e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

SERVER USELESS AND FUNCTION AS A SERVICE

Faiza Anabar A M Dumma*1, Jayashree BS*2, Saqib Ur Rehaman*3

*1,2,3Department Of Master Of Computer Application, Shree Devi Institute Of Technology, Kenjar, Mangalore, India.

DOI : https://www.doi.org/10.56726/IRJMETS60045

ABSTRACT

Server useless computing is a method of providing backend services on an as-used basis. A server useless provider allows users to write and deploy code without the hassle of worrying about the underlying infrastructure.

Keywords: Server Useless Computing, Cloud Computing, Front End, Backend.

I. INTRODUCTION

The beginning of usage of the term 'server useless' can be traced to its original meaning of not using servers and typically was applied to peer-to-peer (P2P) software or client side only solutions . In the cloud context, server useless started to mean that developers do not need to worry about servers and in particular just uses SaaS platforms or services such as Google App 2 Engine. The latest server useless solutions are really server hidden and built to host functions and hide that the functions runs on servers or how scaling is done. The functions may be part of a service (for example Azure Data Lake Analytics or Google Cloud Data lab) or offered as an independent service called Function-as-a-Service or FaaS.

1.1 Problem Statement

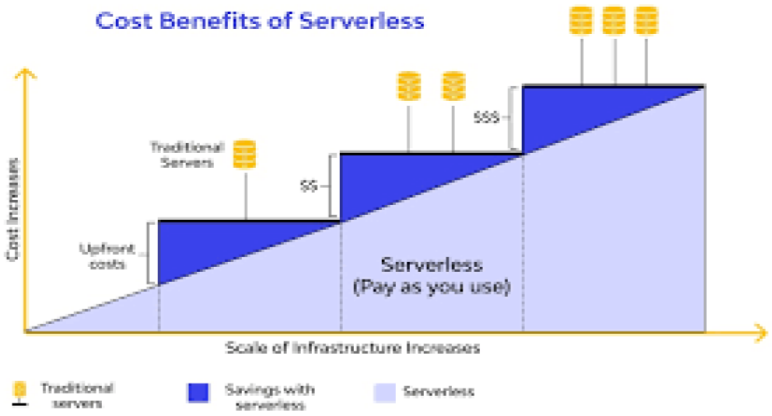
In the early days of the web, anyone who wanted to build a web application had to own the physical hardware required to run a server, which is a cumbersome and expensive undertaking. Then came cloud computing, where fixed numbers of servers or amounts of server space could be rented remotely. Developers and companies who rent these fixed units of server space generally over-purchase to ensure that a spike in traffic or activity will not exceed their monthly limits and break their applications. This means that much of the server space that gets paid for can go to waste. Cloud vendors have introduced autoscaling models to address the issue, but even with auto-scaling an unwanted spike in activity, such as a DDoS Attack, could end up being very expensive.

1.2 Objectives

Server Useless computing and the Function-as-a-Service (FaaS) model are the next stage in cloud computing. They represent a shift away from traditional server-based computing models towards a more dynamic, scalable, and event-driven architecture. Here are some key features and benefits of server useless computing and the FaaS model:

1. No Server Management: With server useless computing, developers do not have to worry about server management, provisioning, or scaling. The cloud provider takes care of these tasks, allowing developers to focus on writing code and building applications.
2. Pay-Per-Use Pricing: Server useless computing follows a pay-per-use pricing model, which means that developers only pay for the resources they use, rather than paying for an entire server or virtual machine.
3. Event-Driven Architecture: Server useless computing is built on an event-driven architecture, which means that applications are triggered by events, such as user actions or system events. This allows for a more dynamic and responsive computing environment.
4. Scalability: Server useless computing is highly scalable, as resources are automatically provisioned and deprovisioned based on demand. This allows applications to scale up or down quickly and efficiently.
5. Faster Development: With server useless computing, developers can build applications more quickly and easily, as they do not have to worry about server infrastructure or deployment.
6. Reduced Costs: Serverless computing can help to reduce costs, as developers only pay for the resources they use, and do not have to worry about managing server infrastructure.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com



II. ARCHITECTURE

Server useless architecture is largely based on a Functions as a Service (FaaS) model that allows cloud platforms to execute code without the need for fully provisioned infrastructure instances. FaaS, also known as Compute as a Service (CaaS), are stateless, server-side functions that are event-driven, scalable, and fully managed by cloud providers. DevOps teams write code that focuses on business logic and then define an event that triggers the function to be executed, such as an HTTP request. The cloud provider then executes the code and sends the results to the web application for users to review. AWS Lambda, Microsoft Azure Functions, Google Cloud Functions and IBM Open Whisk are all well-known examples of server useless services offered by the cloud providers.

The convenience and cost-saving benefits associated with on demand auto-scaling resources, and only paying for services as they're needed, makes server useless frameworks an appealing option for DevOps teams and business stakeholders alike.

➢ Examples of server useless

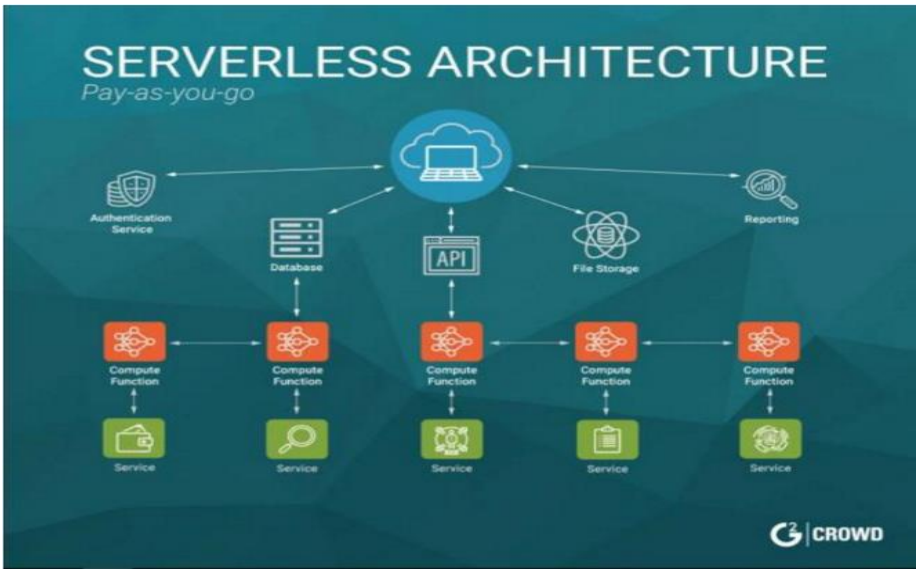
The growing popularity of cloud computing and micro services combined with the demand for greater innovation and agility without increasing costs has contributed significantly to the prevalence of server useless applications.

Notable use cases include:

- **Slack:** Server useless is ideal for independent task based applications such as chatbots and can save on operational costs since billing is based on the actual number of requests. Slack, a popular, cloud-based business communication platform, uses a server useless application called marbot to send notifications from Amazon Web Services (AWS) to DevOps teams through Slack.
- **Home Away:** Reducing development time and server costs while simplifying the build process are goals that universally appeal to business teams and IT teams. Home Away relied on Google Cloud Functions to develop an app that allowed users to search and comment on the recommendations of travelers in real time, even in areas without an internet connection. The cloud service available through Cloud Firestore and Cloud Functions made it possible to set up the infrastructure within minutes and deploy the app within six weeks with just one full-time developer.
- **Green Q:** Garbage pick-up and disposal is an industry that may not seem to require innovative technology, but Green Q took a sophisticated approach to streamlining and improving waste management by using IBM Open Whisk to create an IoT platform that uses hardware installed on garbage trucks to collect key metrics such as pickup time, location, and load weight. The auto-scaling available through server useless was particularly valuable due to the fluctuation of infrastructure demands based on the number of customers and trucks at any given time.
- **Coca-Cola:** Soft drink giant Coca-Cola has enthusiastically embraced server useless after its implementation in vending machines resulted in significant savings. Whenever a beverage is purchased, the payment gateway makes a call to the AWS API Gateway and triggers an AWS Lambda function to complete the transaction. Since vending machines must communicate with headquarters for inventory and marketing

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

purposes, the ability to pay per request rather than operating at full capacity had a substantial impact on reducing costs.



III. FRONTEND VS BACKEND

Application development is generally split into two realms: the frontend and the backend. The frontend is the part of the application that users see and interact with, such as the visual layout. The backend is the part that the user doesn't see; this includes the server where the application's files live and the database where user data and business logic is persisted.

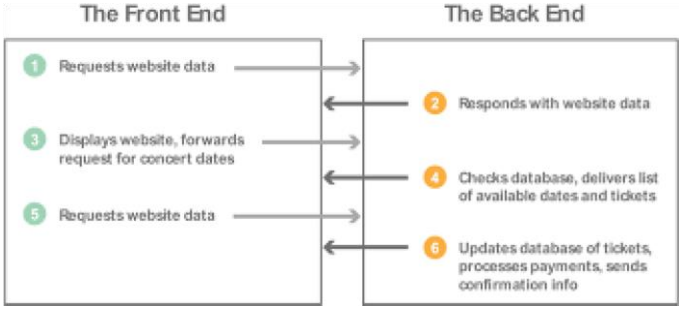


Fig: Frontend vs Backend

For example, let's imagine a website that sells concert tickets. When a user types a website address into the browser window, the browser sends a request to the backend server, which responds with the website data. The user will then see the frontend of the website, which can include content such as text, images, and form fields for the user to fill out. The user can then interact with one of the form fields on the frontend to search for their favourite musical act. When the user clicks on 'submit', this will trigger another request to the backend. The backend code checks its database to see if a performer with this name exists, and if so, when they will be playing next, and how many tickets are available. The backend will then pass that data back to the frontend, and the frontend will display the results in a way that makes sense to the user. Similarly, when the user creates an account and enters financial information to buy the tickets, another back and-forth communication between the frontend and backend will occur.

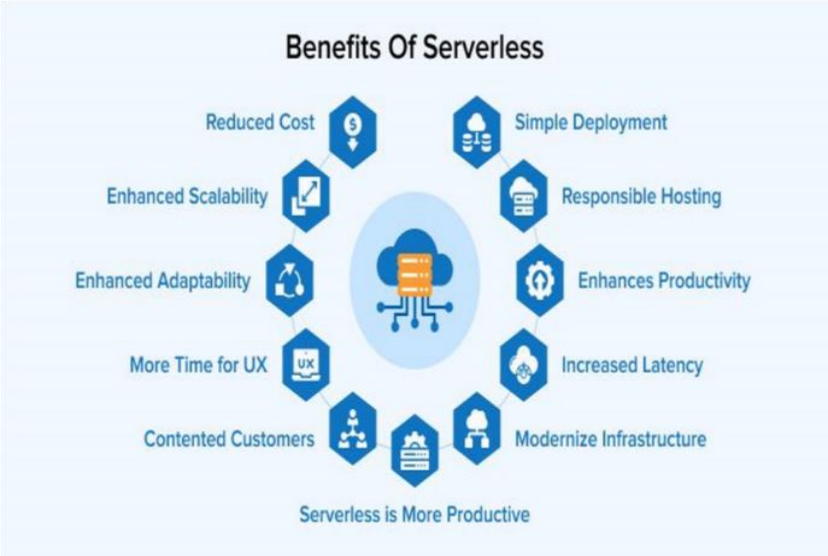
IV. ADVANTAGE AND DISADVANTAGE

Advantage:

- Lower costs – Server useless computing is generally very cost effective, as traditional cloud providers of backend services (server allocation) often result in the user paying for unused space or idle CPU time.
- Simplified scalability - Developers using server useless architecture don't have to worry about policies to scale up their code. The server useless vendor handles all of the scaling on demand.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

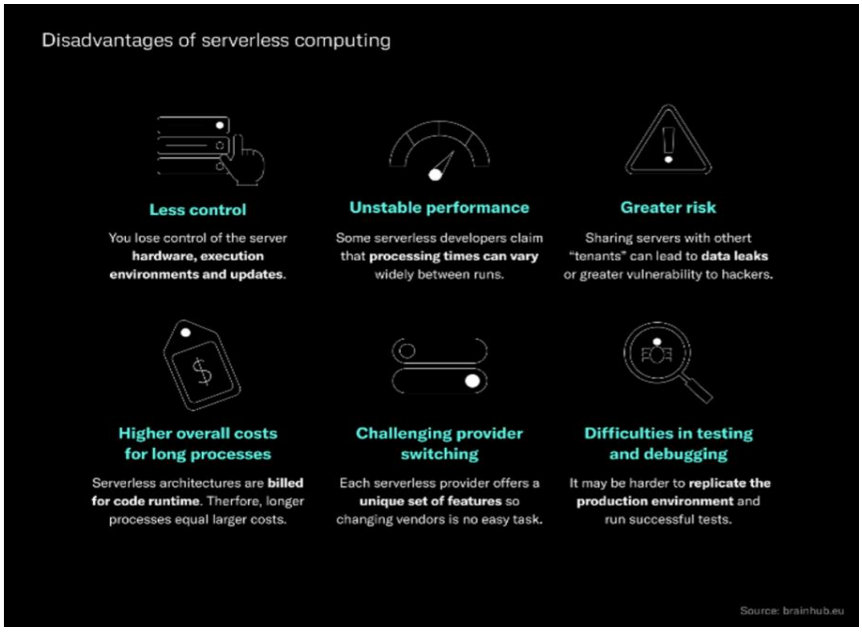
- Simplified backend code - With FaaS, developers can create simple functions that independently perform a single purpose, like making an API call.
- Quicker turnaround – Server useless architecture can significantly cut time to market. Instead of needing a complicated deploy process to roll out bug fixes and new features, developers can add and modify code on a piecemeal basis.



Disadvantage:

- Testing and debugging become more challenging It is difficult to replicate the server useless environment in order to see how code will actually perform once deployed. Debugging is more complicated because developers do not have visibility into backend processes, and because the application is broken up into separate, smaller functions.
- Server useless computing introduces new security concerns when vendors run the entire backend, it may not be possible to fully vet their security, which can especially be a problem for applications that handle personal or sensitive data. Because companies are not assigned their own discrete physical servers, server useless providers will often be running code from several of their customers on a single server at any given time. This issue of sharing machinery with other parties is known as 'multi tenancy' – think of several companies trying to lease and work in a single office at the same time. Multi tenancy can affect application performance and, if the multi-tenant servers are not configured properly, could result in data exposure. Multi tenancy has little to no impact for networks that sandbox functions correctly and have powerful enough infrastructure.
- Server useless architectures are not built for long-running processes. This limits the kinds of applications that can cost-effectively run in a server useless architecture. Because server useless providers charge for the amount of time code is running, it may cost more to run an application with long-running processes in a server useless infrastructure compared to a traditional one.
- Performance may be affected because it's not constantly running, server useless code may need to 'boot up' when it is used. This startup time may degrade performance. However, if a piece of code is used regularly, the server useless provider will keep it ready to be activated – a request for this ready-to-go code is called a 'warm start.' A request for code that hasn't been used in a while is called a 'cold start.'
- Vendor lock-in is a risk allowing a vendor to provide all backend services for an application inevitably increases reliance on that vendor. Setting up a server useless architecture with one vendor can make it difficult to switch vendors if necessary, especially since each vendor offers slightly different features and workflows.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com



How does server useless compare to other cloud backend models?

A couple of technologies that are often conflated with server useless computing are Backend-as-a-Service and Platformas-a-Service. Although they share similarities, these models do not necessarily meet the requirements of server useless.

Backend-as-a-service (BaaS) is a service model where a cloud provider offers backend services such as data storage, so that developers can focus on writing front-end code. But while server useless applications are event-driven and run on the edge, BaaS applications may not meet either of these requirements.

Platform-as-a-service (PaaS) is a model where developers essentially rent all the necessary tools to develop and deploy applications from a cloud provider, including things like operating systems and middleware. However, PaaS applications are not as easily scalable as server useless applications. PaaS also don't necessarily run on the edge and often have a noticeable startup delay that isn't present in server useless applications.

Infrastructure-as-a-service (IaaS) is a catchall term for cloud vendors hosting infrastructure on behalf of their customers. IaaS providers may offer server useless functionality, but the terms are not synonymous.

V. FUTURE SCOPE

Server useless is the next evolution of cloud computing. It provides a pattern for packaging logic while completely abstracting away the underlying environment and administration. This provides beneficial features including on-demand resource scaling and automated infrastructure management, improved efficiency, and pay as you go pricing. Many providers offer great platforms to host functions in a cloud environment. Open-FaaS provides a free solution for teams who want to develop functions that are able to run in Kubernetes, enabling cross-cloud or self-hosted deployments. While powerful with many practical benefits, Server useless really shines in data processing workflows such as realtime event processing, ETL, and web/mobile app backends.

VI. CONCLUSION

To some, server useless and FaaS are the next generation of computing supporting centralized and edge computing with a common event driven programming model. Conversely the drivers of cloud computing are Edge computing and Server useless. One often discusses distributed / edge computing versus centralized approaches and wonders how we move back and forth; the answer is clear -- we have both intrinsically intertwined. Server useless will build the long dreamed infinite limitless computing fabric.

This white paper aims to capture the current state of server useless and FaaS and hopefully inspire a broader community to become involved.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

VII. REFERENCES

[1] https://martinfowler.com/articles/serverless.html
[2] http://lukeangel.co/cross-platform/docker-server useless-faas functions-as- service.
[3] https://hackernoon.com/what-is-server useless-architecture-what are-its pros- and-cons.
[4] https://serverless.com/blog/2018-server useless-community survey huge- growth-usage.
[5] https://server useless.com/framework/docs/providers.