WHAT'S SERVERLESS? PROS, CONS & HOW SERVERLESS COMPUTING WORKS

For a long time, developers were torn. Instead of concentrating on the part of their job that makes the most difference, creating code, they spent a good portion of their time managing and caring for the server infrastructure. Plus, they had to tend to the operating system and web server hosting process required for application.

Their attention divided, maintenance work circular. There had to be an easier way.

Enter serverless computing.

Serverless computing allows developers to build apps without the headache of managing infrastructure. More specially, it enables them to write in serverless code *without* having to:

- Provision a server
- Ensure its functionality
- Create test environments on a server
- Maintain server uptime

This frees up teams and resources to focus their attention on accelerating innovation in today's competitive digital economy.

For teams looking to unburden themselves from server maintenance and wanting to run their apps at scale, serverless computing can be the perfect solution. Which is why it has exploded in

popularity:

- An O'Reilly survey of 1,500 IT professionals found that 40% had adopted serverless architecture.
- Another 2020 survey found that 50% of AWS users were using the serverless function.
- The serverless market is expected to be at $7.7 billion at the end of 2021, up from $1.9 billion in 2016.

Serverless has gone from cutting-edge to the mainstream as developers discover the ease and cost-effectiveness of adopting serverless computing. If you are looking for new ways to deploy and release applications faster and more frequently, then it's time to consider serverless computing.

What is serverless computing?

Serverless computing is a cloud-based service where a cloud provider manages the server. The cloud provider dynamically allots compute storage and resources as needed to execute each line of code.

With serverless computing, the service provider takes care of all the infrastructure (server-side IT), which means all your team needs to do is write code. (Serverless computing is also known as serverless architecture, and it relates closely to functions-as-a-service, or FaaS.)

Technically speaking, servers *are* still involved. However, what makes serverless computing different from architectures we traditionally see in enterprise environments is not a lack of servers—but *how* they are implemented and managed.

Remember that traditional servers have fixed resources that users need to provision for use. On the other hand, serverless computing does not have fixed resources, so it can run multiple applications from a single server.

Developers no longer need to consider the servers for practical purposes. It is all handled for them. The provider takes care of:

- The virtual machine and container management
- Hardware allocating
- Specific tasks built into the code, such as multithreading

Importantly, serverless computing is event driven. Developers create states as I/O requests that are received and then destroyed in compute instances. The process is 100% automated and does not require human interaction and maintenance the way a traditional server would.

This makes serverless computing an efficient, affordable, and resource-effective way to build and use applications.

Beyond the definition of serverless computing, you should understand other considerations and terms about serverless architecture.

Software containers

Software containers power serverless technology. Because the vendor manages them, though, it's invisible to the user. This allows serverless computing to afford the advantages of containerized microservices—without the complexity.

Software containers function as a virtual repository for serverless code. Containers make it easy for developers to write serverless code for more than one platform, like an iOS app and a desktop. Each application has its own container with functionality specific to the platform.

BaaS

Backend-as-a-Service (BaaS) and FaaS are similar in that they both require a third-party service provider dependency. When we think of BaaS, the most well-known example is AWS Lambda.

With Lambda, developers are still using serverless code in software containers, but Amazon works as an intermediary between the user and code implementation in a software container. AWS Lambda will:

- Provides guidelines for developers to follow when submitting code.
- Enter the code into a container via automated processes, providing the entire backend development process as a managed service to their clients.

Other serverless components

The foundation for serverless computing is its serverless architecture. Code designed to work with FaaS or BaaS being offered by third-party vendors powers it.

As far as charges are concerned, these platforms offer a pay-as-you-go structure. Based on runtimes, they charge consumers only for what they use-not for pre-provisioned sets of units.

Thought FaaS and BaaS are popular serverless options, there are a few other serverless architectures of note. These include:

Serverless databases

One drawback of serverless computing for some developers is that it is event-driven and does not have a persistent state. Local variables' values don't persist across instantiations, so it can be a problem for developers who need persistent data.

Serverless databases help fill this critical function for those who want serverless computing but need to store data. Serverless databases function the same as other serverless architecture, and the only key difference is that they store data indefinitely.

Like serverless computing, the job of maintaining and provisioning a database is cared for, and businesses only pay for the compute time used.

(Explore leading serverless databases like MongoDB, Apache Cassandra, and Amazon DynamoDB.)

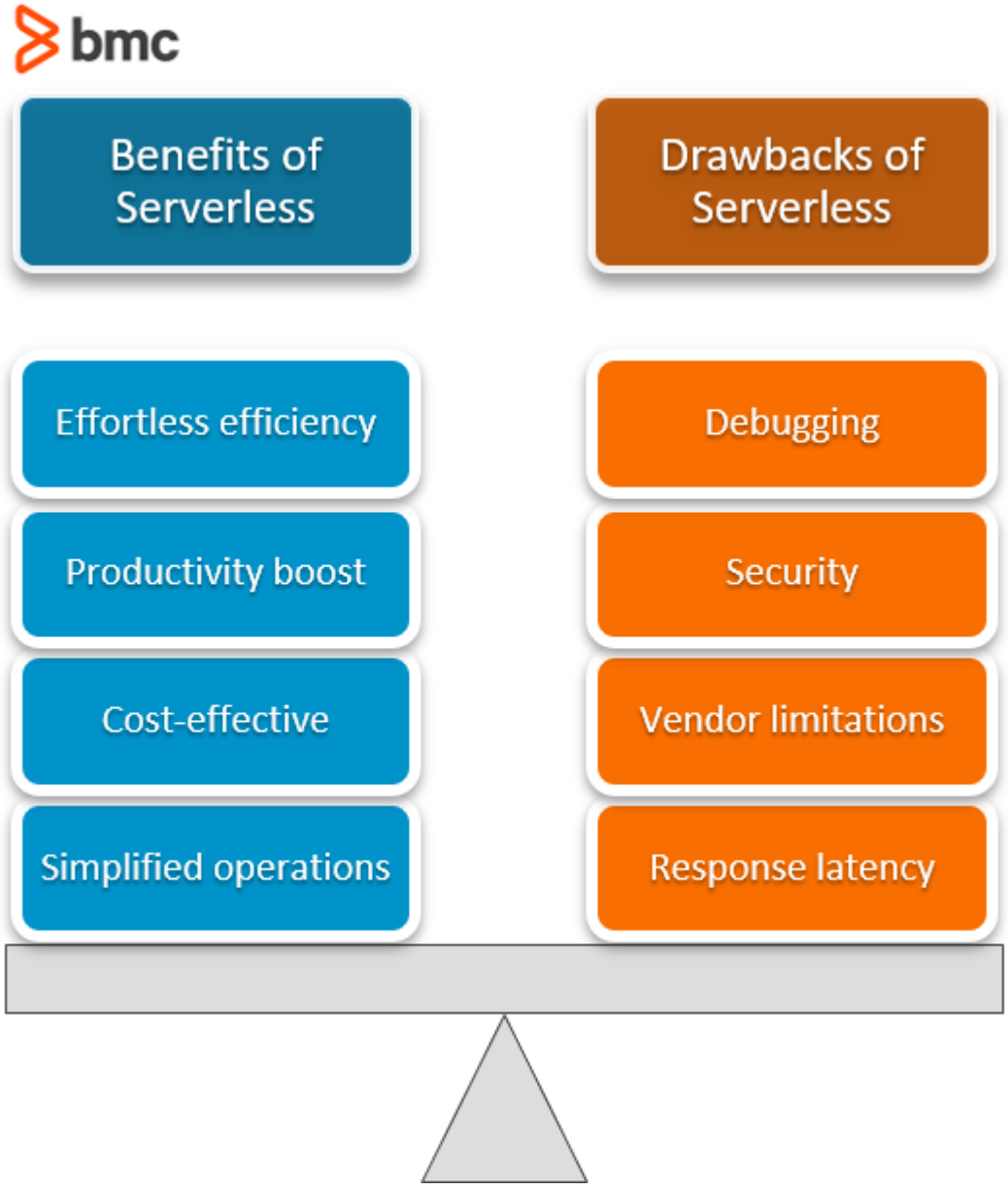
Serverless framework

Serverless frameworks are free, open-sourced software for coding. It builds, compiles and packages code for serverless implementation. It then will deploy it to the cloud.

Developers use serverless framework to simplify and speed the coding process for their teams. It also makes scaling easier across developers and helps to decrease configuration time.

A couple leading serverless frameworks are:

- AWS Serverless Application Model (SAM) consists of template specification and a command line interface (CLI).
- Serverless has both free and paid options.



Benefits of serverless computing

There are several key benefits of serverless computing. Those who make the switch to building applications on serverless platforms can expect these types of benefits and enhancements.

Cost-effective

Switching to serverless computing can have a positive impact on the overall IT budget:

- Without the hardware of a server, maintenance costs are relatively low.
- Labor cost is possibly lower, too, since round-the-clock monitoring and maintenance of servers is not required.
- Serverless computing services charge based on runtimes, so you're never paying for more of a service than you use.

Simplified operations

Since the third-party vendor ensures that server performance meets enterprise demands, operations are simplified.

Companies benefit from allocating resources to other projects and eliminating the need for scalability planning and other strategies that come along with traditional computing models.

Boosts productivity

The nature of cloud computing means that the requests available for end-users are limited to those that are relatively simple to code. With no hardware maintenance requirements, software development teams are free to focus on coding leading to better applications. Serverless computing allows for the rapid building of applications so developers can make the most of their skill sets.

This gives developers more time to work on other projects to enhance a company's business model or service portfolio.

Effortless efficiency

With fewer people, processes, and technologies required to deploy applications, serverless computing offers an efficient way to launch fully scalable applications.

Serverless computing offers:

- Zero server management
- Scaling options
- Optimal availability
- Eliminates idle capacity

Drawbacks of serverless computing

While serverless computing offers numerous benefits, there are some potential downsides for certain developers and teams. Whether these are actual drawbacks for you will depend on the type of product you're building.

Response latency

Serverless computing is an unparalleled framework when it comes to the rapid development of applications. But, how well does the framework perform?

One drawback of switching to serverless computing may be response latency. Response latency is the time between when a request is stimulated and when a program reacts. In the cloud, serverless computing is not continually running—it gets powered down between requests.

As a result, having to start from dormant serverless code can cause response latency. It can take as long as several seconds to spin up and process code. Depending on your specific use case, especially if timing is critical, this could be a point against serverless.

Limitations

Third-party vendors restrict resource limits, which means serverless computing is not ideal for high-computing operations. Even if they were no such limits, certain applications would not be cost-effective from a third-party vendor.

As any cloud service, you'll also want to be aware of vendor lock-in, and what you'd need to do in order to switch serverless providers.

Security

Like anything in technology, you must consider the security factor. When it comes to serverless computing:

- Responsibility of security falls on the servicing company, not the consumer.
- There is a larger attack plain with multiple entry points into the server.

Customers sometimes feel reassured shifting security demands to the company the owns the server, but large servers with multiple entry points may be more vulnerable. And if an attack does happen, consumers are powerless against it, relying instead on the provider to disclose, remediate, and recover.

Debugging

It is generally more challenging to debug serverless code. Devs also have a more challenging time spotting problems as the code doesn't lend itself to digging in for a detailed review of issues or code smells.

Understanding, anticipating, and preparing for these challenges will be critical in meeting the end-user's needs and helping your organization gain a competitive edge with serverless computing.

Serverless computing supports Enterprise DevOps

Including DevOps departments in enterprise organizations makes a bold statement about the status of your digital transformation and capabilities. If you haven't yet transitioned to a DevOps way of working, the time is now.

DevOps is an innovative group that serves as the bridge between development and operations. DevOps teams are well-versed in Agile principles and a variety of coding languages. DevOps developers can build software but spend much of their time churning out patches, instead, to customize and integrate a number of enterprise applications.

Serverless computing is one vital component for any DevOps environments. It might not be right for every dev project, but it certainly can work for many of them.

The serverless era

Every industry, from finance to education to even government, can benefit from serverless computing. This means your employees and clients will be asking for serverless solutions sooner rather than later.

For those looking to build event-based apps quickly and efficiently, serverless computing is the way to conserve resources, increase efficiency, and boost productivity.

Related reading

- BMC DevOps Blog
- The State of Serverless Today
- Microservices vs Serverless: What's The Difference?
- Bring Kubernetes to the Serverless Party
- What Is a Citizen Developer?