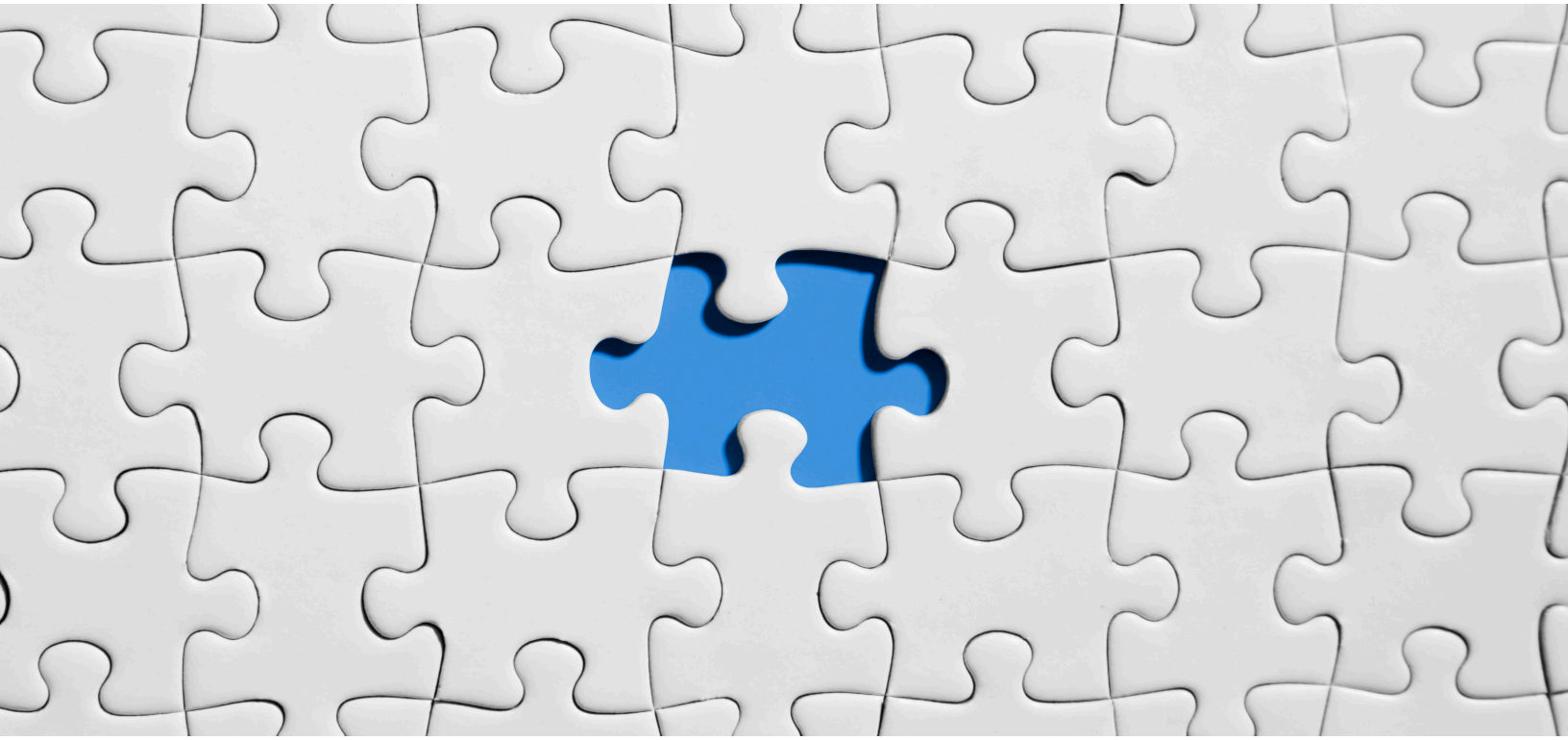
DELL Technologies

ON-PREMISES INFRASTRUCTURE FOR SERVERLESS COMPUTING



Chethan Nagaraj

Sales Engineer Analyst

Dell EMC

Chethan.nagaraj@emc.com

Ravi Sriramulu

Manager 2, Sales Engineer Analyst

Dell EMC

Ravi.sriramulu@dell.com

DELL Technologies Proven Professional

Knowledge Sharing Article

Table of Contents

Introduction 3
Brief of Serverless Computing 3
Backend as a Service (Baas) 3
Function as a Service 3
Serverless vs Others 4
PaaS 4
DevOps 4
Containers 4
Use Case for Serverless Architecture 5
The Benefits and Drawbacks of 'Serverless' 5
Benefits 5
Limitations 5
Hosting Serverless 6
Dell Technologies' Role in Serverless 6
Conclusion 7

Introduction

Serverless architecture, a new buzzword in IT jargon, refers to running applications without servers. Or is it? While applications indeed need servers on which to run their code, the major difference is how the applications are written or rewritten to run the code in a way that frees the application owner from thinking about running, provisioning, configuring and maintaining servers.

Technologies such as Platform as a Service (PaaS), Containers and DevOps aim to free application development from server configuration. So what does a Serverless platform bring to the table and how different is it from the methodologies mentioned above? In this Knowledge Sharing article, we explain how Serverless architecture combines Function as a Service (FaaS) and Backend as a Service (BaaS), among other things and the business sectors it caters to.

We will also discuss today's implementation of the service from Public cloud providers and the inherent issues that arise from it, i.e. vendor lock-in, security, optimization and drawbacks from the serverless architecture such as start-up latency, testing, deployment, versioning, migration, and so on.

Finally, we will discuss on how Dell Technologies can cater to the infrastructure needs of Serverless platform and overcome the drawbacks that the technology has today.

Brief of Serverless Computing

Serverless computing is running applications with zero awareness of underlying physical infrastructure. This enables developers to concentrate on developing applications rather than spend time thinking about the physical configuration and its limitations. There are other computing architectures which work more or less the same, such as PasS, DevOps, and Containers. The differences between these and Serverless will be explored in the next section. While all of them try to achieve physical infrastructure abstraction, Serverless computing takes this approach to the next level.

There are two major of models of serverless computing:

Backend as a Service

Typically used to develop applications for Web and Mobile, Backend as a Service (BaaS) relies heavily on the use of third party applications, databases and APIs to manage server-side logic. Web apps and Mobile apps are difficult to scale as their user base is very dynamic, an app can go viral any time. Developing on BaaS platform places the onus of scaling resources with sudden rise in traffic on the service provider.

Function as a Service

In this architecture, applications contain stateless, event-triggered bits of code with a certain amount of server-side logic. Again, the compute/server infrastructure is entirely managed by a third-party service provider. These work well for high volume transactions, where each transaction is isolated and serviced by an instance of a function. Scaling is very simple for application developers as multiple instances of a function can be initiated to handle multiple requests based on events.

Serverless vs Others

PaaS

In PaaS, most applications need to be brought up and down for each request, whereas FaaS does something different. The big difference is in how you operate your app.

The key operational difference between PaaS and FaaS is that PaaS providers such as Heroku or Openshift usually lack in scaling feature. A user of traditional PaaS application needs to define the amount of resources for the applications. It's possible to manually scale the application up and down per usage by changing the number of resources assigned but mostly it is the responsibility of a developer or administrator.

Also, in PaaS – designed for long-running applications – it is expected that the applications are always available to serve user requests. Clearly, bringing down the server for application scaling would not meet that expectation. Whereas in FaaS function, user request is served and the function is terminated once the request is processed. It would start again once there is another request and the cycle repeats.

Your functions should not consume provider resources when there are no incoming requests.

DevOps

DevOps – a combination of development and operations – may have to be rethought because serverless is changing how software is being built and operated.

In recent years, those who develop the applications also host it on the machines. Both tend to be distinct. Recently, virtual machines have replaced physical machines and containers. A person creating a code are not the one overseeing the Kubernetes clusters, managing the data center in the wee hours.

DevOps has becoming the new standard in the IT industry, widely adopted, quite often in conjunction with containers and cloud technologies. However, at the same time organizations are struggling to fully utilize its features, perhaps due to organizational challenges and shortage of expertise.

Serverless doesn't mean 'no ops' but it might mean 'no system admin' depending on how far we take it forward. First, 'Ops' means a lot of work on administrative tasks as well as monitoring, networking, and ample amount of debugging and system scaling. These problems also exist in Serverless architecture and we need to still deal with it. Second, sys admin is still happening; the only difference is that we are outsourcing it with Serverless.

Containers

Another main reason for Serverless FaaS is that we avoid managing the computational processes at the operating system level. Another similar popular abstraction of processes are containers, a popular example of such technology is Dockers. We have also come across various container hosting systems such as Mesos and Kubernetes which abstract individual applications for OS-level deployment. Further, cloud container platforms such as Amazon, ECS and Google Container Engine like Serverless FaaS as it enables their teams to avoid managing their own server systems. Given these examples, how different is FaaS Serverless computing?

If the gap of management and scaling between Serverless FaaS and hosted containers narrows, the choice between them may come down to style and type of application being used. FaaS would a better fit if an application is event-driven style with few event types per application component whereas containers are considered a better choice for synchronous-request driven components with multiple entry points.

There is an inherent similarity between containers and serverless computing. For example, Kubernetes – an open source container orchestration platform based on the application provided metrics – provides an automated scaling method. AWS lambda actually runs a container for each function.

Container and containers orchestration may provide an efficient way to implement FaaS and provide an additional abstraction level that hides specific processes, operating system and containers from the developers allowing them to focus more on coding.

Use Case for Serverless Architecture

One of simplest and direct applications of serverless architecture is REST APIs. Serverless APIs can be built quickly using the serverless frameworks. Building REST APIs is not difficult, all one needs is a basic web framework, a glue code which can 'talk' with the backend and a library for rendering the data in whichever format you are returning.

Serverless framework helps in performing all the common functions such as auto-scaling in REST API and benefit from 'Pay as you use' model and results in less cost for hosting application. Serverless platform plays a major role in building varied applications that can be integrated seamlessly with the cognitive intelligence and data analytics. Also, IOT devices and cognitive chat bots perform well on serverless platforms.

Benefits and Drawbacks of 'Serverless'

Benefits

- Lowers development and operational costs.
- Less expensive to scale: no need for developers to newly implement the code to scale nor administrators to upgrade existing servers or add servers.
- Reduces software complexity.
- Simplifies software packaging and its deployment, requiring no system administration.
- Works with agile development, enabling developers to focus on code and deliver faster.
- Reduces time-to-market and accelerates software release.

Limitations

- Not efficient for long-running applications: In certain cases, using long running tasks can be much more expensive.
- Vendor lock-in: Your application is completely dependent on third-party provider, thus don't have full control over the application. Also, dependence on platform availability and platform's API and costs can change as per the needs.
- Takes time to handle a first request by your function: This problem is known as 'Cloud Start' – a platform needs to initialize the internal resources (AWS lambda, for example needs to start a container). The platform would release the resources if there have been no activities on the

functions for a long time. This can be avoided by sending periodic requests to your functions to ensure the functions remain in an active state.

- Architecture Complexity: There should be a balance between how many functions an application may call. It gets cumbersome to manage too many functions and ignoring the granularity creates mini-monoliths in the end.
- AWS Lambda limits the number of concurrent executions that can be running of all your Lambdas. There is a limit for each AWS account. Some organizations might use single account for both production and testing. For instance, organizations doing a new type of load tests and starts trying to execute 1,000 concurrent Lambda functions would unknowingly get Denial of Service (DoS) for their production applications.
- Implementation Drawbacks: Integrating testing serverless app is tough. When compared with other architectures, the unit of functions with Serverless FaaS are much smaller and hence, there is a lot more reliance on integration testing than with other architecture styles. You may also face issues related to deployment, versioning and packaging. An entire logical application may require deploying a FaaS artifact separately.

Hosting Serverless

AWS is one of the first to offer platform for serverless computing though AWS Lambda. Microsoft and Google, the other two market leaders in cloud, followed suit and began offering their own platforms for serverless computing, named Microsoft Azure Functions and Google Cloud Functions, respectively.

Customers are implementing their serverless applications in one of the cloud platforms listed above. This is very helpful when in the following scenarios:

- To launch new applications without worrying about infrastructure needs
- Usage of these applications is very low
- During testing phase

Small businesses and startups find it financially viable to go with cloud offerings rather than host their —— own infrastructure, especially when there is a sudden rise in the usage or when the application goes 'viral'. However, once gaining a steady user base with high volumes, the economics tend to turn the other way and the cost of hosting serverless applications on a public cloud suddenly no longer makes sense.

Dell Technologies' Role in Serverless

This is the right time for Dell Technologies to offer services in serverless computing. Their current offering – using Elastic Cloud Storage as backend storage for Amazon S3 – is limited, involving using an open source Serverless platform for functions and front-ending tasks and integrating it with S3 platform and then to the ECS. This is a very complicated implementation and defeats serverless architectures' purpose – simplicity.

There is a need for a turnkey solution in the Dell Technologies product portfolio. The best way to achieve it is by using the latest version of Pivotal Cloud Foundry platform and integrate it with ECS. This would result in an easy-to-deploy infrastructure with minimal management overhead. More importantly, with PCF's serverless platform, application developers can continue to concentrate on

application development alone. At the same time, the organization will have complete control of the data and its security – all achieved at much lower cost compared to hosting the application in the cloud.

Conclusion

In this article, we have provided a basic understanding of serverless computing, compared different types, and presented benefits and limitations. Similar to Containers and DevOps, Serverless computing is a major revolution in software development. Developers everywhere are embracing the serverless architecture, as they discover they no longer have to write the codes to address infrastructure maintenance and deal with infrastructure woes.

With the help of Elastic Cloud Storage and Pivotal Cloud Foundry's serverless platform, all the benefits of serverless computing can be brought to on-premises infrastructure, greatly reducing the cost and minimizing security risks of hosting critical applications in cloud.

Reference

Mike Roberts (May, 2018) Serverless Architectures https://martinfowler.com/articles/serverless.html