A Primer on Serverless Computing

By, Janakiram MSV

The shift to cloud computing fundamentally changed the way software is built and consumed by developers. Along with core infrastructure, higher-level components of the technology stack started to become available as services. The rise of APIs and mobile computing prompted cloud providers to deliver Platform as a Service (PaaS) and Backend as a Service (BaaS). Developers targeting next-generation experiences through the web, mobile, and wearable applications started to consume these services by letting the cloud providers manage the mundane but critical aspects of software such as compute, storage, networking, management, security, and scaling.

The Shift to Serverless

During the last few years, a new paradigm of computing started to get the attention of developers - serverless computing. This trend is considered to be the fourth wave of computing where x86 servers, virtual machines, and containers represented the first three generations. In serverless computing, developers squarely focus on the code and not on the underlying infrastructure. They need not plan the number of servers, amount of storage, and the network topology of deployments. Developers write code snippets in their favorite language and directly upload them to a serverless computing platform for execution. Multiple such code snippets or functions are logically connected to form a complex application. Since the platform deals with one function at a time, and functions are the fundamental deployment units, this model is often called as Functions as a Service (FaaS).

FaaS is fundamentally different from Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). In IaaS, customers are expected to spin up virtual machines to deploy binaries of their application. When dealing with PaaS, developers deploy a codebase of an entire application to the cloud platform. When targeting FaaS, developers don't think of a whole application. They only write, test, and deploy one function at a time. These functions are assembled together to deliver a unique feature or functionality of an application. So, a function is to FaaS what a VM is to IaaS.

There are three critical attributes to a serverless platform -

- Per-second billing based on the execution time
- Transparent resource provisioning through infrastructure abstraction
- Event-driven invocation

Unlike virtual machines and containers, functions deployed in FaaS are billed only for the execution time. That means when an external application invokes the function, only the duration of execution is considered for billing. Most of the functions running in FaaS are designed to be stateless and short-lived processes. This innovative pricing model makes serverless computing very attractive for certain classes of applications.

Prior to the deployment or execution of a function, developers need not provision or configure infrastructure. The FaaS provider handles the technology stack comprised of servers, operating systems, runtimes, libraries, frameworks, and core dependencies of the environment. FaaS will also handle resource scaling transparently without the developer or ops team performing scale-in and scale-out operations.

Finally, functions deployed in serverless environments are invoked by internal or external events. They are expected to respond to a new message in a queue; an HTTP request routed via the API gateway; an IoT device changing its state, and much more. Functions communicate to each other via events. This encourages a loosely-coupled, event-driven design of applications.

Customers benefit from FaaS due to its fine-grained deployment model and per-second pricing model. Apart from reuse of code, FaaS drives better utilization of resources. That's one of the reasons why web-scale companies such as Expedia and Netflix are embracing Serverless computing.

While enterprises with complex requirements may not be refactoring applications for FaaS, they can leverage them for increasing the efficiency of deployments. Serverless computing environments can invoke code in response to events or in regular time intervals. Like Expedia, enterprise customers can use FaaS to verify and validate backup sets of applications. As soon as a backup file is written to an object storage bucket, it can be verified to ensure accuracy. Similarly, the log files generated by the applications and infrastructure can be processed by FaaS in real-time to find anomalies and unusual usage patterns.

Web applications can quickly add new functionality through FaaS. For example, search, and translation functions can be implemented in FaaS while easily integrating them with existing web frontends.

Serverless Market Landscape

There are a few commercially available serverless computing platforms in the market. All the major cloud providers are investing in FaaS delivery model. Here is a quick summary of the available choices:

AWS Lambda

Announced at AWS re:Invent 2014, this is one of the first, most mature, and stable serverless frameworks available in the market. Started with Node.js, this service now supports multiple languages including .NET, Java, and Python. Over a dozen AWS services are integrated with Lambda, and the list is only growing. Mobile and IoT developers love Lambda due to the power and flexibility it brings.

Azure Functions

Microsoft is not far behind in offering a serverless computing platform. Azure has all the required building blocks to deliver a FaaS environment. Developers looking at implementing a serverless solution can consider using Azure Functions, the runtime environment for invoking code snippets either on-demand or through a scheduled Cron job. Microsoft is expanding its serverless environment by adding event-driven messaging through Azure Event Grid.

Google Cloud Functions

Google is at the forefront of the microservices paradigm. Apart from driving containerization, the company is investing in an AWS Lambda competitor called Cloud Functions, which will run on its public cloud infrastructure. Still in the initial preview stage, Google Cloud Functions are hosted as containers running on Google Compute Engine. The platform only supports Node.js but is expected to add additional languages in the future. Only a few services such as Google Cloud Storage and Google Cloud Pub/Sub are currently integrated with Cloud Functions. Given the wide footprint Google APIs have, the platform would support multiple other services such as Gmail, Cloud Messaging, Maps, and others.

IBM Cloud Functions

Announced at the IBM InterConnect event in 2016, OpenWhisk is an open source alternative to AWS Lambda. IBM has donated OpenWhisk to Apache, which is now an incubated project. Apart from supporting Node.js, OpenWhisk can run snippets written in Swift. Developers can package functions written in any language packaged as Docker containers that can be invoked by OpenWhisk. Commercially available as Cloud Functions, this service is integrated with IBM Bluemix, the PaaS environment powered by CloudFoundry.

Serverless Computing and DreamFactory

DreamFactory Services Platform comes with embedded PHP and Node.js runtime environments. As the API-driven middleware for a variety of backend data sources, DreamFactory provides a powerful mechanism to manipulate the HTTP request and response pipeline. Developers can take control of the processing pipeline by writing server-side code to perform tasks at various stages.

For example, to prevent someone from creating or deleting an object, developers can use the pre_process handler to inspect at the request and raise an exception. If they need to change the response in a specific way, they can use the post_process handler to modify the payload.

Server-side scripting in DreamFactory can be used to create database-agnostic triggers. DreamFactory provides a way for a script to call other services in the environment. This enables the implementation of very flexible workflow triggers. Based on a field value or other condition, developers can send an email, create an object, or trigger a push notification. They can also call external services such as SendMail, Twilio, or Salesforce to easily integrate data.

Since server-side scripting in itself is exposed as a REST API endpoint, developers can easily deploy code snippets in DreamFactory. They can use cURL or any other REST client to dynamically manage the versioning and invocation of functions.

With MQTT becoming a first-class citizen in DreamFactory, server-side scripting becomes a powerful serverless feature for developers. They can create a set of rules that can send commands to a variety of connected devices. This feature effectively becomes a dynamic, serverless rules engine for DreamFactory developers. This concept

was demonstrated in the MQTT tutorial that controls Raspberry Pi devices connected to DreamFactory.

Summary & Key Takeaways

Serverless computing, despite the confusing name, is a paradigm where developers rely on an environment for executing short-lived code snippets. It is a logical evolution of Backend as a Service (BaaS), where developers consume services directly in the frontend. Functions as a Service (FaaS), moves server-side code from long running components to ephemeral function instances.

Serverless computing complements existing delivery models such as IaaS, BaaS, and PaaS instead of replacing them. Customers running their workloads in IaaS or PaaS can easily extend their use cases to FaaS to bring new functionality.

All the major public cloud vendors are investing in serverless computing platforms. AWS, Azure, Google Cloud Platform, and IBM Bluemix have mature serverless offerings.

DreamFactory developers can take advantage of server-side scripting to achieve the benefits of serverless computing. The integration of MQTT and Push Notifications make it an ideal event-driven execution environment.