Serverless Computing

Introduction

Serverless computing is a type of cloud computing that reduces the burden of managing infrastructure to the cloud provider. A developer uploads their code into a Serverless platform (often based on the popular Docker or Kubernetes containers) where the code sits and waits to be triggered by an external event. Once the code has completed it is no longer left running. The developer or operations teams do not need to manage the environment. The Serverless platform provider is responsible to ensure that there are sufficient computing resources to handle spikes in usage including processing power and storage. The Serverless computing provider deals with patching and all aspects of infrastructure management so that the developer does not have to. Serverless lends itself well to small discrete portions of code known as *functions* that perform a short lived defined task (often lasting less than five minutes). Another significant benefit of adopting Serverless Computing is that you pay for what you consume often in the cents rather than paying for a predefined volume of server resources. Serverless computing has become popular by the adoption of Amazons Lambda and the Microsoft Azure Functions environments.

Definitions

Event-driven architecture (EDA) is a software component, or event, which executes a response to receiving one or more event notifications (or triggers).

Serverless computing is a type of cloud computing where the customer does not have to provision servers for code to run on. The cloud provider starts and stops a container platform as a service as requests are triggered. Consumption is often billed on a per second usage model.

Function as a service (FaaS) refers to cloud services that enable serverless app development and management. FaaS users are able to program and deploy their code without having to manage their own server(s). Code is triggered by remote events (such as by a mobile app) with code execution occurring in the Serverless environment rather than on the end user device.

Docker container is an open source software development platform that provides developer/operations (Devops) staff to package applications in **containers.** Containers provide application portability between different application hosting providers that support applications built for the Docker platform. Unlike with Serverless computing environments the use of containers such as Docker require that users still must ensure that their application remains up and running.

Endpoint security is an information security methodology for protecting network connected devices (endpoints) by monitoring their status, activities, software, authorization and authentication. Security software is installed on an endpoint device. In a Serverless computing environment the API which is the interface to the FaaS service represents an endpoint and is often secured by the use of Security software (such as antivirus, antispyware, firewalls, intrusion detection).

Government Use Cases

Governments are adopting new software development methodologies and frameworks as they migrate traditional computing environments to cloud first and cloud native platforms. Many government systems contain information processing workflows that are replicated in dedicated applications to process citizen social services application forms, or for permits such as land usage or building development applications. By adopting a function as a service methodology Government developers could access a library of open source functions that could be reused or modified to meet developer's needs rather than to have to duplicate the same business functions over and over again. Possible examples include:

Citizen Feedback Forms Many web services provide a citizen response form where information is collected at the click on a citizen web page, The information is then relayed as a stream of data to the server where a function can process the information or trigger another Serverless function to prioritize the information for alerting a government agent. /1/

Extract, Transform and Load (ETL) data processing: Many government applications are data driven where a dedicated ETL product is being licenced or run on a dedicated server to extract data from a source, transform all or a portion of the data and then store data into a database. By adopting a Serverless approach to this task the application can be transformed to a utility based model where the function is only run when required and the cost of operation is reduced to the actual execution time for the ETL job. /2/

Image Processing: Functions provided by Azure can trigger on actions embedded in a stream of data and perform analysis on the information; this could find application in GIS environments.

Mobile and Internet of Things applications: Authentication services developed using Serverless developed functions can ensure that a user or a device is authorized to access information or invoke a service /3/ .

Standards

Currently no environment specific standards around the usage of serverless computing have been adopted by the international or domestic standards bodies such as the ISO, IEEE or NIST. Adherence to recognized standards and best practices concerning security, data transfer, data storage and data sovereignty must be considered when adopting serverless computing. Refer to resources presented at the end of this document.

Security and Privacy Considerations

The benefits of adopting Serverless computing can be significant when migrating from a traditional capital and operating cost model to a cloud native pay as you go model. IT spend is rationalized by shifting from a 7/24 execution environment where servers have to be allocated, managed, patched, secured to the serverless model where code is executed on demand and costs can be reduced to a fraction of the traditional billing model.

Threats still exist in the form of a **DDoS** attack, or **SQL injection** attacks. The developer must **validate inputs**, use good code libraries and best practices. Many Serverless functions will invoke other data handling apps that offer data storage and retrieval. Developers need a **trustworthy data handling framework**.

When selecting a Serverless application provider investigate and understand their **API gateway** which will handle the traffic between the user and your function code. Investigate how to apply security on this communication.

As with most software applications only give access to those who require it. Provide the **minimum of permissions** to users or roles. Do not put keys in client applications. Understand the **key management** requirements for your application environment. Work with your security team to ensure that your solution is sustainable.

Since Function as a Server/Serverless computing is based on event driven architecture understand what the **event triggers** are, what their security requirements will be and implement a security strategy that minimizes communication to only those system parts requiring a notification.

Understand potential **privacy implications** embedded in business logic. Serverless functions are basic building blocks that should not expose privacy related information. Utilize a privacy checklist and run scenarios through your user stories.

Recommendations

- Work within the BCDevExchange and OCIO DevOps communities to learn, experiment and test potential use cases for modern cloud technologies such as Serverless. Share your experiences.
- Understand your business requirements when developing replacement IT solutions and leverage shared function capabilities rather than developing narrow business specific solutions where applicable.
- Take a function portfolio approach to your IT services, leverage reusability where practical.
- Understand how your functions can be invoked by unintended systems, ensure you have adequate security built into your solution.
- Build a log access review process into your service lifecycle.
- Functions are nano-services and still require a security assessment.
- Ensure that your developers are familiar with the OCIO ISB Cloud Security Framework.
- Perform a Privacy Impact Assessment (PIA) for your Serverless implementation.

Summary

Serverless Computing, often referred to as *Function as a Service* eliminates the need to manage, provision and scale the computing platform. The burden of ensuring 7/24 operation is shifted to the Serverless computing platform provider. You only pay for what you use plus a nominal subscription fee, thereby significantly reducing the operating costs for an application. However Serverless computing is not always the best fit for all application development. Understanding best use case scenarios and the shifted operational responsibilities from the infrastructure to the application will help government IT departments to realize the benefits of elastic cloud computing. DevOps teams will focus more on the Dev side than on the Ops. Ops teams can take on new roles in advising developers on security and function portfolio expectations.

References:

/1/ https://aws.amazon.com/blogs/publicsector/using-a-serverless-architecture-to-collect-and-prioritize-citizen-feedback/
/2/ https://engblog.nextdoor.com/bender-ff65a6edee92
/3/ https://docs.aws.amazon.com/lambda/latest/dg/with-on-demand-custom-android.html

Resources:

1. Deloitte (2018) : Serverless Computing –Architectural Considerations & Principles https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology-media-telecommunications/Serverless%20Computing.pdf
2. OpenFAAS : https://www.openfaas.com/
3. Demystifying Serverless and OpenFaaS http://collabnix.com/demystifying-openfaas-simplifying-serverless-computing/
4. Serverless for Beginners https://acloud.guru/learn/serverless-for-beginners
5. Amazon AWS Lambda https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
6. Microsoft Azure Functions https://azure.microsoft.com/en-us/services/functions/
7. ISO Standards https://www.iso.org/standards.html
8. IEEE Standards http://standards.ieee.org/findstds/index.html
9. Cloud Adoption Security Framework, Province of BC, OCIO (February 2017)
10. Containers vs Serverless Computing https://rancher.com/containers-vs-serverless-computing/