

**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

**Universidad del Perú, Decana de América**

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

MENCIÓN INGENIERÍA DE SOFTWARE

|  |  |
| --- | --- |
| **Nombre del estudiante** | Gonzales Flores , Alejandra Gonzales  Vega Inga , Antony |
| **Nombre del Tema** | Arquitectura de Software |
| **Título del Tema** | FaaS , Function as a Service |
| **Nombre del Tutor** | Félix Melchor Santos López |
| **Fecha de vencimiento** | Sábado , 13 de julio de 2019 |
| **Fecha de Presentación** | Sábado , 13 de julio de 2019 |
| **Cantidad de Palabras** |  |

PLAGIARISM

Plagiarism is the presentation by a student of an assignment which has in fact been copied in whole or in part from another student's work, or from any other source(e.g. published books, periodicals, or the web) without due acknowledgement in the text.

COLLUSION

Collusion is the presentation by a student of an assignment as his or her own which is in fact the result in whole or in part of unauthorized collaboration with another person or persons.

Before submitting my assignment, I have:

1. Made a copy of the assignment and of any material submitted with the assignment.

2. Ensured that my assignment and any material submitted are clearly identified.

3. Retained a copy of the email submission of this assignment (if appropriate).

4. Attached all files and required material to the email (if appropriate).

Declaration

I declare that this assignment is my own work and does not involve plagiarism or collusion. I also declare that the material contained in this assignment has not previously been submitted for assessment in any other formal course of study.

|  |  |
| --- | --- |
| FaaS Function as a Service | |
| Alejandra Paola , Gonzales Flores | Antony, Vega Inga |

Abstract

Keywords: Function as a Service , serverless computing, cloud-native applications

**Introducción**

Esta revisión inicia con el contexto de las aplicaciones cloud-native , a continuación se revisa el concepto de serverless computing ,y funciones como servicio o Function as a Service (FaaS) describiendo su concepto , beneficios y tradeoffs . A continuación listaremos los principales proveedores de tecnología FaaS . Luego se listaran casos de uso donde se aplicó FaaS. Finalmente se describen las conclusiones de esta revisión.

**Aplicaciones Nativas de la nube (*Cloud-Native Applications* )**

Durante esta revisión bibliográfica no se ha encontrado una definición general del concepto “aplicación *cloud-native*” . (Gannon, et al., 2017) enuncian que son aquellas aplicaciones tienen generalmente las siguientes propiedades (i) operan globalmente,(ii) deben escalar bien con miles de usuarios , (iii) están construidas bajo la suposición que la infraestructura es fluida y que la falla es constante , (iv) están diseñadas de modo que la actualización y pruebas ocurren sin interrumpir la operación en producción y (v) la seguridad no es una preocupación tardía. (Gilbert, 2018) enuncia que las aplicaciones *cloud-native* cumplen con tres promesas velocidad, seguridad y escalabilidad por medio de los siguientes características: (i) utilizan infraestructura desechable, (ii) esta compuesta por componentes aislados pero unidos, (iii) escala globalmente,(iv) aprovechan los servicios en la nube, y (v) utilizan servicios de diferentes proveedores.

Para el desarrollo de aplicaciones *cloud-native* existen servicios y estos a su vez son *cloud-native*. (Asghar, et al., 2018) describen la evolución de los servicios *cloud-native* en términos de virtualización tal como se muestra en la Fig. 1. El color negro se muestra el grado de virtualización que es manejado por el proveedor, el color plomo representa las plataformas usadas por los desarrolladores para el despliegue del código en el ambiente virtualizado, el color rojo representa el desperdicio de recursos y el color blanco representa los recursos controlados por el desarrollador .En su inicios las aplicaciones eran desplegadas en directamente en el hardware del servidor (A). Esto implicaba un gran desperdicio de recursos y una dificultad para la escalabilidad. También implica que el desarrollador tenía otras preocupaciones además del aplicativo. En(B) se muestra la virtualización de hardware mediante el uso de máquinas virtuales. En(C) se muestra se muestra la virtualización de sistema operativo por medio de contenedores. Tanto en B y C se mejora la escalabilidad y el manejo de los recursos y una reducción en las preocupaciones del desarrollador. Finalmente (D) muestra la virtualización del tiempo de ejecución por medio de Función como servicio (FaaS)

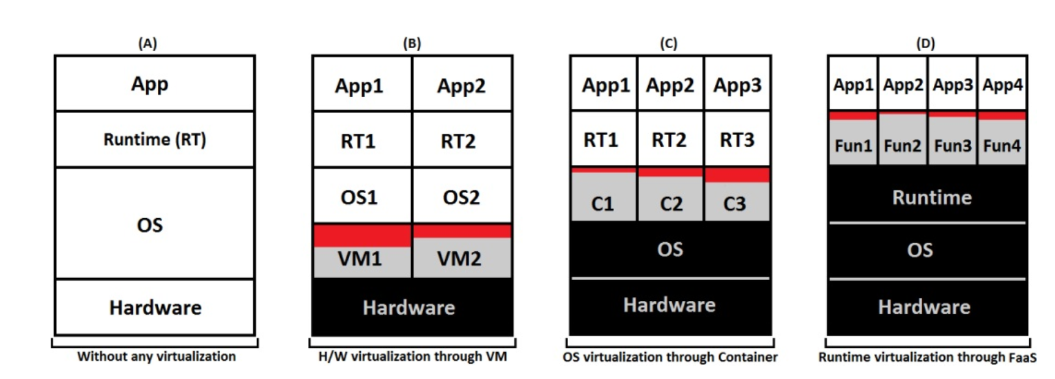


Fig. 1: Evolución hacia los servicios cloud-native

**Computación sin servidor (Serverless Computing) y Funciones como servicio Function as a Service**

(van Eyk, et al., 2018) definen *Serverless Computing* como “*una forma de cloud-computing que permite a los usuarios ejecutar aplicaciones guiadas por eventos y pagadas granularmente, sin tener que preocuparse por la lógica de operacional*“. (Roberts, 2018) describe que el concepto Serverless abarca dos grandes áreas superpuestas (i) Aplicaciones que incorporan de forma significativa o completa aplicaciones y servicios de terceros alojados en la nube, para administrar la lógica y el estado del servidor. (ii) Aplicaciones donde la lógica del lado del servidor todavía está escrita por el desarrollador, pero se ejecuta en contenedores sin estado, son activadas por eventos, son efímeras (pueden durar solo una invocación) y son administradas completamente por un tercero. Un ejemplo de estas aplicaciones son las Función como servicio o FaaS

(van Eyk, et al., 2018) definen Función como servicio o FaaS como “*una forma de serverless computing donde el proveedor de la nube maneja los servicios , ciclo de vida y ejecución guiada por eventos de las funciones proveídas por el usuario* ”. (Roberts, 2018) describe las siguiente características (i) FaaS permite ejecuta código de back-end sin la necesidad de administrar servidores propios o servidores en la nube como PaaS. Al subir el código de nuestra función a un proveedor de FaaS, el proveedor es el encargado del aprovisionamiento de recursos, la creación de instancias de máquinas virtuales, la gestión de procesos, etc. La escala horizontal es completamente automática, elástica y gestionada por el proveedor (ii) FaaS no requiere desarrollar en una librería o framework especifico ya que existe una gran variedad de lenguajes disponibles en los diferentes proveedores de FaaS. (iii) Las funciones en FaaS generalmente son activadas por los tipos de eventos definidos por el proveedor.

(Roberts, 2018) lista tres puntos importantes a tomar en cuenta al desarrollar una función FaaS. Primero las funciones de FaaS tienen restricciones significativas cuando se trata del estado local (máquina / enlazado a instancia) ya que no garantiza que dicho estado persista en múltiples invocaciones. Por ello, las funciones FaaS a menudo se describen como sin estado, pero es más exacto decir que cualquier estado de una función FaaS que deba ser persistente debe externalizarse fuera de la instancia de la función FaaS. El segundo punto es la duración ya funciones de FaaS suelen estar limitadas en cuanto al tiempo que se permite ejecutar cada invocación. Esta duración es establecida por el proveedor. El tercer punto se refiere al tiempo de inicio y el arranque en frio es decir el tiempo para que la plataforma FaaS inicialice una instancia de una función antes de cada evento. Esta latencia de inicio puede variar desde unos pocos milisegundos hasta varios segundos

# Trabajos citados

Akidau, T., S. C. & Reuven, L., 2018. *Streaming systems : the what, where, when, and how of large-scale data processing.* Sebastopol, CA: O'Reilly, 2018.

Asghar, T. y otros, 2018. Feasibility of Serverless Cloud Services for Disaster Management Information Systems. *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS),* pp. 1054-1057.

CNCF, C. N. C. F., 2018. *CNCF Cloud Native Definition v1.0.* [En línea]   
Available at: https://github.com/cncf/toc/blob/master/DEFINITION.md  
[Último acceso: Julio 2019].

Gannon, D., Barga, R. & Sundaresan, N., 2017. Cloud-Native Applications. *IEEE Cloud Computing,* 4(5), pp. 16-21.

Gilbert, J., 2018. *Cloud native development patterns and best practices : practical architectural patterns for building modern, distributed cloud-native systems..* Birmingham, UK: Packt Publishing, 2018.

Maas, G. & François, G., 2019. *Stream Processing with Apache Spark.* Sebastopol, CA: O'Reilly Media.

Microsoft Azure, s.f. *Build cloud-native applications in Azure.* [En línea]   
Available at: https://azure.microsoft.com/en-us/overview/cloudnative/  
[Último acceso: Julio 2019].

Newman, S., s.f. *Serverless Fundamentals for Microservices: An Introduction to Core Concepts and Best Practices,* s.l.: O'Reilly Media, Inc. 2018.

Roberts, M., 2018. *Serverless Architectures.* [En línea]   
Available at: https://martinfowler.com/articles/serverless.html  
[Último acceso: Julio 2019].

van Eyk, E. y otros, 2018. Serverless is More: From PaaS to Present Cloud Computing. *IEEE Internet Computing,* 22(5), pp. 8-17.