Asian Journal of Research in Computer Science

Volume 16, Issue 4, Page 194-210, 2023; Article no.AJRCOS.107654
ISSN: 2581-8260

The Rise of Serverless Architectures: Security Challenges and Best Practices

Burak Cinar [a*]

Think Secure Inc., United States of America.

Author's contribution

The sole author designed, analyzed, interpreted and prepared the manuscript.

Article Information

DOI: 10.9734/AJRCOS/2023/v16i4382

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://www.sdiarticle5.com/review-history/107654

Review Article

Received: 12/08/2023
Accepted: 17/10/2023
Published: 25/10/2023

ABSTRACT

The field of serverless computing has had significant growth and recognition in the past decade. This emerging area has garnered attention because to its notable impact on cost reduction, latency reduction, scalability improvement, and elimination of server-side management, among other benefits. Nevertheless, there is still a dearth of comprehensive study that would facilitate developers and academics in gaining a more profound comprehension of the importance of serverless computing in many scenarios. Therefore, it is imperative to provide scholarly study data that has been published within this particular field. This study conducted a comprehensive analysis of 275 scholarly articles retrieved from reputable literature sources, with the aim of extracting valuable insights pertaining to serverless computing. Subsequently, the acquired data underwent analysis in order to address many study inquiries pertaining to the contemporary advancements in serverless computing, encompassing its fundamental principles, available platforms, and patterns of utilization, among other relevant aspects. In addition, we analyze the current obstacles confronting serverless computing and explore potential avenues for future research to facilitate its deployment and utilization.

Keywords: Serverless; computing; serverside management; virtualization; cloud.

**Corresponding author: E-mail: burak@thinksecure.io;*

1. INTRODUCTION

The utilization of virtualization technology has been crucial in facilitating the widespread acceptance and accomplishments of cloud computing [1, 2]. Cloud providers enabled the concurrent sharing of resources with multiple users by encapsulating their monolithic applications within virtual machines (VMs). This approach ensured robust isolation assurances while affording users seemingly limitless resources that were promptly accessible whenever their applications required them [3]. The aforementioned characteristics, in conjunction with a pay-per-use economic model that has resulted in decreased total cost of ownership for cloud consumers, have positioned cloud computing as the most prosperous computing paradigm of the previous decade [4]. Nevertheless, this achievement was not without its disadvantages, with the primary downside being the requirement for users to personally oversee the virtual machines (VMs) [5, 6].

The aforementioned matter has prompted the observation of novel programming models that have significantly altered the methodologies employed by software developers in the creation and administration of cloud-based services [7, 8]. A programming approach that is often used involves the decomposition of an application into numerous components, referred to as microservices [9, 10]. These microservices are autonomous, have limited scope, and are loosely connected. They are able to communicate with each other via standard APIs. Regrettably, the inefficiency of virtual machines (VMs) in terms of prolonged startup time and substantial resource consumption has been demonstrated in their application for executing microservices [11, 12]. Consequently, other container technologies, such as Docker, were proposed as a more lightweight alternative [13, 14]. Containers have several advantages over virtual machines (VMs) in terms of enhanced portability, reduced start-up time, and improved resource usage [15, 16]. These benefits contribute to the streamlining of the development and administration processes for large-scale applications deployed in cloud environments. The aforementioned benefits have prompted cloud providers to embrace container technologies and integrate them with orchestration systems such as Kubernetes or Docker Swarm [17, 18]. This integration enables the seamless automation of deploying, scaling, and managing cloud-based applications that are built on microservices. Nevertheless, akin to the utilization of virtual machines (VMs), the microservices paradigm necessitates users to undertake the configuration and administration of the underlying containers, encompassing associated libraries and software dependencies [19, 20]. Furthermore, it relies on a static billing model, wherein users are charged a predetermined amount for the allocated resources, irrespective of the actual resources consumed. The aforementioned problems make microservices inappropriate for some categories of applications [21].

The concept of serverless computing is gaining prominence as a novel computing paradigm for the deployment of applications within cloud environments [22, 23]. The current iteration possesses two significant advantages in comparison to its previous versions. Primarily cloud computing enables software developers to delegate infrastructure maintenance and operational responsibilities to cloud providers so enabling them to concentrate exclusively on the core business logic of their programs [24]. Additionally, the platform operates on a pay-per-use framework, wherein users are solely billed in accordance with the amount of resources they utilize [25]. At now, serverless computing is available in two distinct variations, namely backend as a service (BaaS) and function as a service (FaaS) [26]. The fundamental concept underlying Backend as a Service (BaaS) is to offer software developers a range of services and tools, such as databases, APIs, file storage, and push notifications, with the aim of facilitating and expediting the process of developing mobile and online apps [27, 28]. Function-as-a-Service (FaaS) primarily centers around enabling software developers to deploy and run their own functions on cloud infrastructure [29, 30]. It is important to note that these functions can also leverage supplementary services, similar to those provided in Backend-as-a-Service (BaaS) offerings. Currently, Function as a Service (FaaS) is widely recognized as the prevailing serverless paradigm. Throughout the remainder of this essay, the term "serverless" will be employed as a reference to Function as a Service (FaaS) [31, 32].

Serverless computing is becoming increasingly popular in the business as an appealing approach for deploying applications and services in the cloud, mostly due to its simplicity and economic benefits [33, 34]. Cloud service providers, such as Amazon, Microsoft, Google, IBM, and Alibaba, have already been providing serverless computing services to its clientele. Likewise, other corporations, including Netflix, T-Mobile, and Realtor, are currently experiencing the advantages of serverless computing [35]. Based on recent industry surveys, it is projected that the serverless computing market would experience a compound annual growth rate (CAGR) of 26% from 2020 to 2029 [36]. Nevertheless, as the frequency and variety of assaults targeting cloud systems continue to rise, it becomes evident that ensuring security and preserving privacy will play a crucial role. Failure to adequately address these concerns has the potential to impede the general acceptance and implementation of serverless computing [37-39].

1.1 What is Serverless Security?

Serverless security refers to a protective layer that is specifically designed to safeguard code functionalities. The technology is implemented directly into the apps, allowing developers to impose compliance measures in order to increase the security posture. However, in order to comprehend its importance, it is necessary to take a moment to delve into the concept of serverless architecture [40, 41].

1.2 Serverless Architecture

Serverless architecture refers to a software development methodology in which the design and execution of an application are conducted without the need for concern for the underlying infrastructure. The responsibilities of your team will be limited to the tasks of writing and executing the code, while the cloud provider will play a facilitating role in managing the application's servers. During the initial stages of software development, the deployment of programs involved the utilization of bare metal servers, which were overseen by system administrators. The resource allocation for the project was limited, resulting in significant costs. Nevertheless, advancements in cloud computing, virtual machines, and containerized applications have facilitated the development of applications with enhanced flexibility, simplicity, and efficiency [42, 43]. Serverless computing might be likened to the subsequent installment within a series of films. The primary objective of servers is to enable and enhance the communication and collaboration between users and various applications. Despite being necessary, servers introduce a significant level of complexity, need more IT operations management, and incur expenses. In contrast, the implementation of a serverless architecture allows developers to prioritize the development of high-quality code rather than dedicating resources to server maintenance, backup creation, and security measures. The cost-effectiveness of this approach lies in the fact that users are only charged for the specific services they utilize, and these services are only utilized during the operation of the program [44, 45].

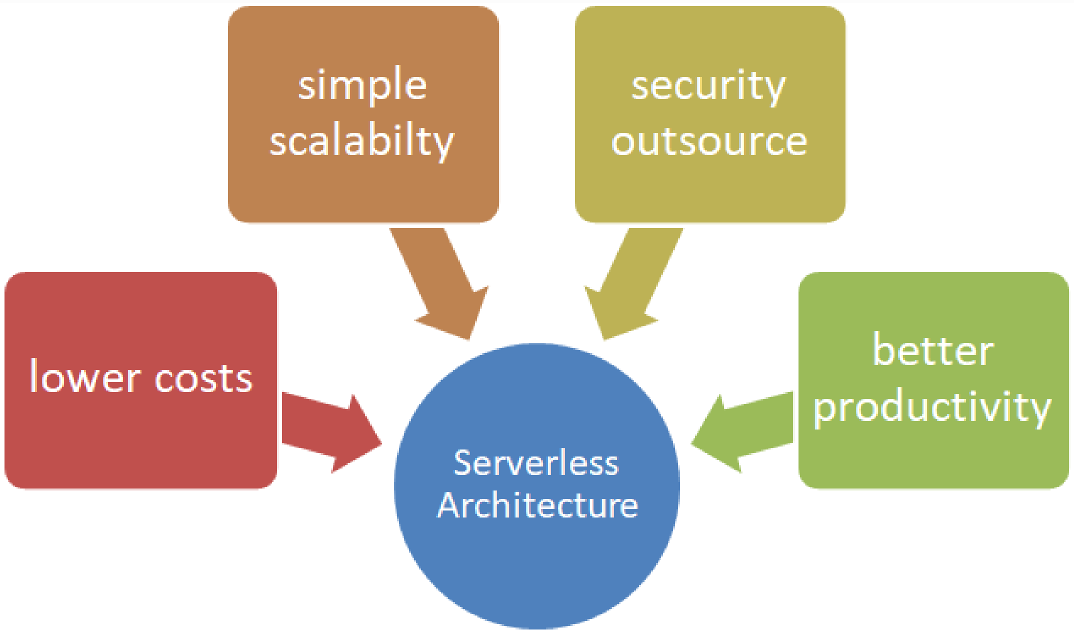

Fig. 1. Serverless architecture

A serverless architecture, such as Function as a Service (FaaS), enables the development of code in the form of discrete functions that execute in response to specific events [46]. However, it should be noted that the transfer of security responsibilities to the cloud service provider is advantageous. The utilization of serverless architecture facilitates the automation of workflow processes, resulting in several advantages such as enhanced scalability, expedited application delivery, and decreased development expenditures [47, 48].

1.3 What are the Benefits of Serverless Architecture?

The serverless architecture is a methodology that operates on events rather than streams, hence enhancing its resilience to faults. When a failure occurs in the program, it has a localized effect on the single event rather than affecting the entire log [49, 50]. There exist five additional advantages associated with the utilization of serverless architecture. By engaging in the practice of outsourcing server and database management, organizations can effectively decrease the financial burden associated with employing human resources to oversee infrastructure and computing space [51, 52]. By entrusting the cloud provider with the security control of your infrastructure, you effectively implement measures such as runtime security, key and secrets management, and automated patching, in accordance with established best practices [53, 54]. With the implementation of security safeguards by providers such as Azure, AWS, and Google Cloud, the application code is encompassed by fundamental policies [55]. Application containers are rendered less vulnerable to attacks due to their termination upon cessation of active operation [56, 57]. The absence of a state creates an inherent security stance. The utilization of serverless architecture enables the decomposition of applications into smaller modules, facilitating efficient tracking and monitoring of serverless applications. The provision of IAM (Identity and Access Management) for each individual function confers enhanced security measures [58, 59].

2. WHAT ARE THE CHALLENGES OF SERVERLESS ARCHITECTURE?

It’s not all rosy with serverless computing and it has some challenges too.

2.1 Security Misconfigurations

Cloud service providers offer a variety of security measures and settings; nonetheless, it is imperative to ensure their proper configuration. Omitting or misconfiguring any element in a given context can potentially result in a risk [60, 61].

2.2 Improper Permission Privileges

One potential drawback associated with granting individual access to numerous operations is the possibility of inadvertently providing a user with excessive rights beyond what is actually required. It is imperative to consistently employ the practice of implementing the least privilege or zero level permits in order to effectively mitigate the potential for attacks [62].

2.3 Event-Data Injections

It is possible for untrusted inputs to be injected into the functions whenever an event is triggered. As a result, you need to thoroughly evaluate each event source to check for unauthorized data injections [63, 64].

2.4 Verbose Error Messages

By ignoring verbose error signals such as "out of memory," "null pointer," and a multitude of other failures, as well as improperly handling exceptions, hackers can find a weakness in the system that they can exploit and use to launch an attack [65, 66].

2.5 Third-Party Vulnerabilities

The burden of protecting the application will need to be shared between the cloud providers and the developers in order to combat the vulnerabilities that come with database services, backend cloud services, configurations that are related with the application, and so on [67].

2.6 Serverless Best Practices for any Cloud

The characteristics of a serverless architecture that contribute to its allure are also the characteristics that make it less secure. For instance, while the numerous functionalities of apps make it possible to implement fine-grained security policies, this also means that the number of entry points that might be targeted by attackers is increased. Implementing the best practices for serverless security is necessary in order to protect your application from being attacked [68, 69].

Through the use of layered access control and authentication, you can protect against unauthorized application access. Your cloud provider will present you with a variety of options to choose from in order to mitigate the danger posed by faulty authentication. OAuth, SAML, and OpenID Contact are some of the tools that are available. You can create and implement a password difficulty system that is tailored to your organization's development culture in conjunction with a multi-factor authentication system [70, 71].

It is essential to have effective monitoring and recording of user access and the runtime of your functions in order to reduce your vulnerability to security assaults. Even while the capabilities of observability and monitoring are provided by your cloud provider, it is still a good idea to invest in a third-party solution that provides you with monitoring-specific features that make the experience more natural [72].

Implement the principle of "least privileged access" by establishing a strict permission policy and delegating one-of-a-kind roles to each individual function. Developers have a tendency to over-privilege, which results in a security flaw that attackers love to take advantage of. This is because the chore of giving authorization access to every function is a challenging one. Your development and security teams should meet face to face and have a conversation about the goals of each function and the precautions they need to take to ensure its safety. This is the best practice [73-76].

Implementing the appropriate policies for code analysis will allow you to maintain control over your functions and ensure that you will not deploy any code that has bugs. Because hostile actors are more likely to target personnel than application code, it is vital to build security controls that check that every function that is pushed through to the continuous integration and continuous delivery pipeline is clean and does not contain any dangerous elements [77, 78].

Set a ticking clock for each of your functions so that they are stopped as soon as the corresponding task is complete. This will ensure that your functions have sufficient runtime. You may eliminate the possibility of malicious code being injected into your application by using serverless function timeouts to remove any window of vulnerability that may exist. You should also pay attention to the runtime that you assign. Developers have a tendency to go with the maximum duration available, which means that attackers get more time to do something malicious. Paying attention to the runtime that you assign is important[79, 80].

In order to avoid being dependent on a third party, you should implement a rigorous process to check the originality and dependability of your sources. In addition, if you want the safest code possible at that time, you should make sure that you are using the most recent version of each component that comes from an open-source code. When employing open-source components, one of the most important best practices is to ensure that they are always kept up to date [81-83].

When it comes to managing secrets effectively, you need to pay careful attention to sensitive credentials like API keys. It is a best practice to include period evaluation in configuration files or to make use of a secrets scanning tool such as Spectral to automate this procedure for you. Either option is acceptable [84, 85].

Implement security not only during the testing phase of the SDLC but throughout the entire process. You will be able to cut down on operational costs and minimize delays if you include security at every level of the development process. In addition, the ongoing examination will point out any security flaws and areas that require the implementation of stringent protection procedures [86, 87].

2.7 What Is Serverless Computing?

Serverless computing is a novel paradigm in which application developers do not need to maintain servers. Instead, they deploy code as functions, and servers are assigned based on demand. This frees application developers from the burden of managing servers. It makes use of the Function-as-a-Service (FaaS) architecture, which is a sort of cloud computing that enables programmers to easily package and distribute their code without having to deal with the necessary server infrastructure. An event-driven computing execution architecture is what FaaS is. In this architecture, developers design logic that is then deployed in containers that are fully controlled by a platform, and the logic is then done on demand [88, 89].

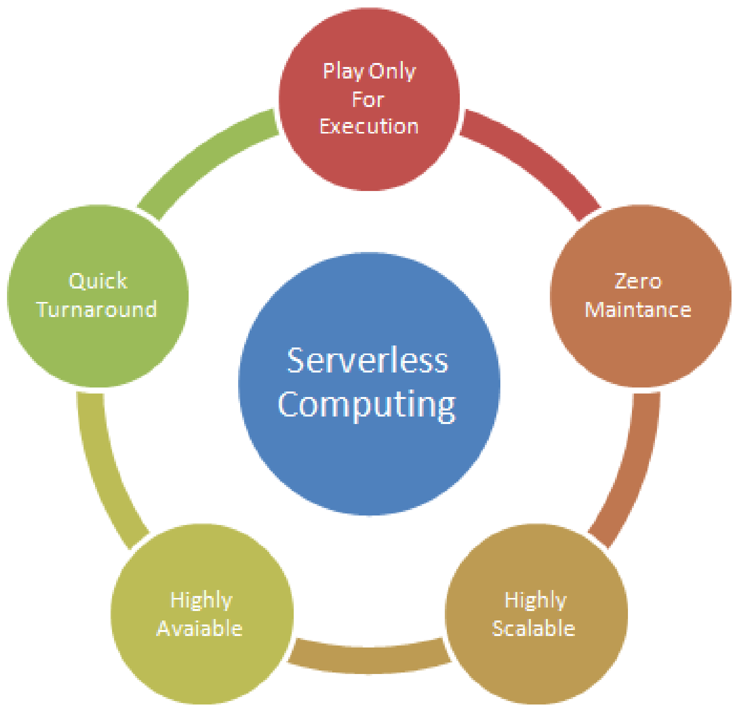

Fig. 2. A journey towards Serverless computing

The servers are abstracted away from the application development process in serverless computing, and the duty for provisioning, maintaining, and scaling the server infrastructure is taken on by the cloud provider in response to the code events that are deployed. Once they have been launched, serverless programs automatically adjust to demand and scale up and down in response to changing requirements. Serverless products offered by public cloud providers are frequently subjected to event-driven execution and on-demand metered pricing models. Because of this, the use of a serverless function does not incur any expenses when it is not being utilized. A database, user authentication, a web server, a security token service (STS), and Lambda functions are some of the components that might make up an example of a serverless system solution. The most well-known instances of Functions as a Service are provided by Google Cloud Functions, Microsoft Azure Functions, and Amazon Lambda respectively [90, 91].

The capacity of serverless technology to speed up the process of software development is one of the primary reasons why its implementation has become more widespread in recent years. It makes it possible for developers to delegate the maintenance of server infrastructure to a Cloud Service Provider (CSP), who is then responsible for taking care of the application functionality. However, the fact that the cloud service provider (CSP) is only responsible for the security of the cloud and not the security in the cloud is the most significant challenge for serverless architectures. This indicates that the serverless application is not only remains vulnerable to the dangers and flaws that are experienced by traditional programs, but it is also subject to the security difficulties that are specific to the design of serverless applications. Developers of serverless applications need to take responsibility for their apps by implementing identity and access management (IAM), resource configuration, and the protection of code functions and libraries [92, 93].

3. SERVERLESS SECURITY RISKS

3.1 Increased Attack Surfaces

The input data that serverless functions use comes from a wide number of event sources. These sources include HTTP APIs, cloud storage, IoT device connections, and queues. This considerably expands the surface area that might be exploited by attackers, as some of these components may contain untrusted communication formats that the typical application layer protection would not be able to thoroughly examine. If the separate vulnerabilities of the connection links that are utilized to fetch input data (such as protocols, vectors, and functions), can be exploited, then those connection linkages could be used as points of attack [94, 95].

3.2 Security Misconfiguration

Serverless apps are vulnerable to cyber assaults because cloud service providers often give insecure setups in the settings and functionalities they provide for their customers. For instance, denial-of-service assaults, also known as DoS attacks, frequently take place in serverless applications due to incorrectly specified timeout settings between the functions and the host. In these attacks, the low concurrent limitations are used as entry points into the program to carry out the attack. Attackers are also able to take advantage of the function linkages by interjecting the function calls, which causes the function events to take far longer to complete than anticipated. This opens the door for Denial-of-Wallet (DoW) attacks and drives up the cost of the serverless function. DoW attacks are also caused by the leakage of sensitive data, which can occur when unprotected functionalities from public repositories (such as GitHub and S3 buckets) are used. This is due to the fact that attackers take use of functions that are available to the public that include unprotected secrets and keys that are hardcoded in the code [96-98].

3.3 Broken Authentication

Serverless apps do not store state information, and the use of microservices in their architecture leaves the various movable pieces of the independent processes vulnerable to authentication failure. For instance, in an application that contains hundreds of serverless functions, if the authentication for just one of those functions is handled incorrectly, it will have repercussions for the remainder of the program. Attackers could zero in on a single function to get access to the system using a variety of ways, including automated brute force attacks and dictionary attacks [99-101].

3.4 The Threat of Over-Privileged Functions

The serverless ecosystem is dependent on a large number of autonomous services, and each of these tasks has its own set of responsibilities and permissions. The significant amount of interaction that takes place between functions might occasionally result in functions being overprivileged in their rights. For example, due to the fact that actors are able to see it, a function that continuously consults the database and changes other functions could represent a significant security concern [102, 103].

4. SERVERLESS SECURITY BEST PRACTICES

4.1 Use API Gateways as Security Buffers

Using API HTTPS endpoint gateways is one strategy for preventing event-data injection in serverless apps. This strategy involves separating data from functions. An application programming interface (API) gateway will serve as a security buffer due to the fact that data will be retrieved from a wide variety of sources. This separation between app users on the client-side and serverless services on the backend will be created by the API gateway. This decreases the area that can be exploited by an attacker by offering multiple security checks through the use of a reverse proxy. When you use HTTPs endpoints, you are able to exploit inherent security protocols, such as data encryption and the key management provided by your provider. These protocols are beneficial since they assist in protecting the sensitive data, environment variables, and stored data [104, 105].

4.2 Data Separation and Secure Configurations

You should implement preventative measures, such as code scanning, the separation of commands and queries, and the identification of any exposed secret keys or unlinked triggers, and then configure these measures such that they correspond to the CSP's best practices for serverless applications. Doing so will help you avoid denial of service attacks. To prevent execution calls from being disrupted by DoS attackers, function timeouts should be reduced to their bare minimum [106].

4.3 Dealing with Insecure Authentication

Implementing numerous specialized access control and authentication services is required if you want to reduce the likelihood of your authentication being compromised. You can make authentication more difficult to circumvent by utilizing the access control solutions provided by the CSP. These options include OAuth, OIDC, SAML, OpenID Connect, and multi-factor authentication (also known as MFA). Additionally, you have the ability to implement unique password complexity criteria and restrictions with regard to length and character type, which makes it difficult for hackers to crack your passwords [107, 108].

4.4 Sufficient Serverless Monitoring and Logging

You will need to make an investment in a comprehensive observability and monitoring solution in order to obtain in-depth visibility into all of the functions contained within a serverless application. Because it does not cover the application layer, relying exclusively on the logging and monitoring capabilities supplied by the CSP is not enough to ensure adequate security. The application event data that is contained within it is vulnerable to security breaches, and if it is not constantly watched, it could serve as a potential entry point for assaults. This is a significant danger that cannot be ignored [109, 110].

4.5 Minimize Privileges

The separation of functions from one another and the limitation of the interactions between them through the use of IAM roles provisioned on their rights is the recommended method for reducing the number of privileges in independently operating functions. This could also be accomplished by ensuring that the program is executed with the bare minimum number of permissions necessary to carry out an event correctly [111, 112].

4.6 Separate Application Development Environments

The concept of separating the multiple environments into staging, development, and production is one of the most effective ways to provide continuous development, integration, and deployment (CI/CD), which is one of the finest development practices. This guarantees that effective vulnerability management is prioritized at each and every level of the development process before moving on to the next version of the code. Additionally, it ensures ongoing testing and improvement of the program through patch priority, safeguarding updates, and finding vulnerabilities, which enables developers to keep one step ahead of attackers [113, 114].

5. STAYING AHEAD OF SERVERLESS SECURITY RISKS WITH SYSDIG

It is impossible to overstate the level of popularity that abstraction has among cloud agent models due to the growing adoption of cloud platforms and serverless architecture. Within the serverless environment, the utilization of virtual images and containers (such EKS and ECS) as host machines is continually expanding. Container security, on the other hand, is the most difficult aspect of the containerization process. Container security refers to the process of ensuring that security protocols are applied in order to safeguard the underlying infrastructure, runtime, and data in container applications [115]. Falco was developed by Sysdig in order to facilitate the acceleration of innovation and the increase of standardization in the Container-as-a-Service (Caas) industry. Falco assists in the detection of threats across containers, cloud-native hosts, and Kubernetes. In addition, Sysdig has developed serverless agents with the assistance of AWS Fargate to make the Container-as-a-Service model more user-friendly and simpler to monitor for the occurrence of security events within the containers. This was accomplished. The entirety of Sysdig's product catalog will assist in resolving the majority of the serverless security issues outlined above and will guide you through the process of implementing the ideal procedures for a flawless serverless solution [116].

Guidelines that should be followed in order to improve the safety of serverless applications

5.1 Ensure the Confidentiality of Sensitive Information

Encrypt every piece of data and utilize a trusted storage method for your credentials. Conduct a review of the roles and permissions that have been assigned to the various users, third parties, and application operations. In addition to this, you need make custom roles according to the requirements and assign those roles to the functions.

5.2 Include a Plan for Handling Incidents in Your System

It is absolutely necessary to put an incident response plan into action in order to be able to

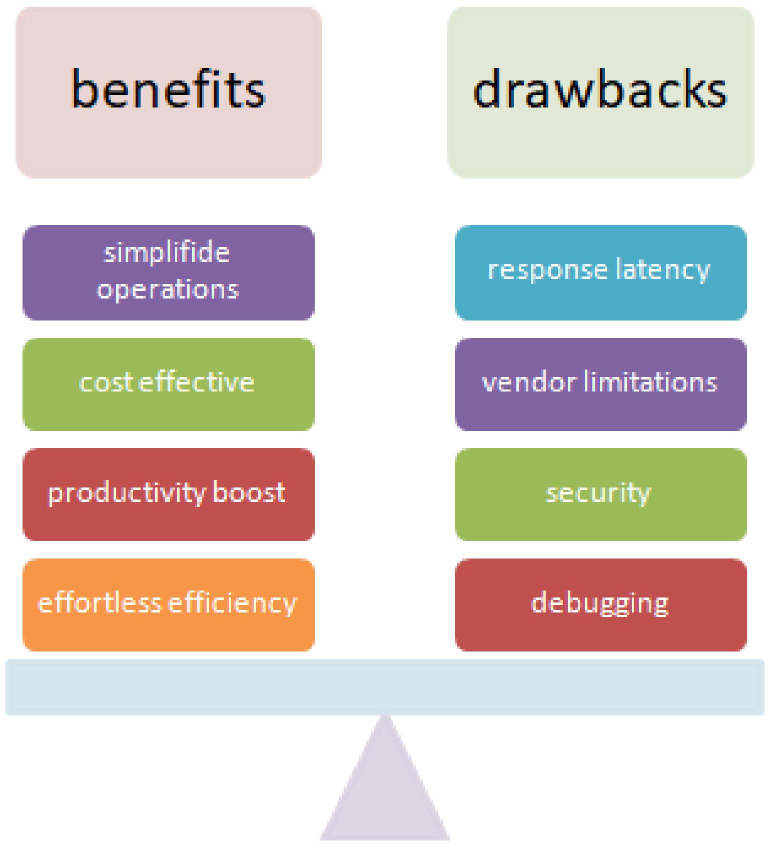

Fig. 3. Serverless pros and cons

recognize the early warning indications of an assault. This helps to uncover concerns earlier so that they can be resolved in accordance with their severity. This ensures that the application is protected.

5.3 Ensure that Appropriate Security Logs are Kept

Your application's level of protection relies heavily on its ability to maintain accurate monitoring and security logging. The vendors of cloud services each have their own set of recommendations to solve this problem.

5.4 Dependencies on Outside Parties

It is advisable to get rid of dependencies that aren't necessary. Priority should be given to the ongoing monitoring and upgrading of the framework's, libraries', and other dependencies' versions, as well as the creation of security patches for earlier versions of the dependencies' frameworks and libraries [117].

6. CONCLUSION

The process of deploying and delivering software has continued to advance, particularly ever since the introduction of reasonably priced and dependable cloud hosting. There has been an explosive development in the number of businesses "going serverless," which is a terrific approach to create scalable apps thanks to the serverless designs that provide this great way. Having said that, additional caution is necessary. Due to the fact that this model of providing security is based on shared responsibility, all parties involved should be aware of what it is that they are responsible for securing.

COMPETING INTERESTS

Author has declared that no competing interests exist.

REFERENCES

1. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D,

Rabkin A, Stoica I, Zaharia M Above the clouds: A berkeley view of cloud computing. Tech Rep, University of California at Berkeley; 2009.

Available:http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html

2. AWS Lambda; 2021.

   Available:https://aws.amazon.com/lambda/

   Accessed 21 Oct 2022.

3. Azure Serverless | Microsoft Azure;2021. Available:https://azure.microsoft.com/solutions/serverless/.

   Accessed 21 Oct 2022.

4. Serverless Computing Solutions—Google Cloud; 2021. Available:https://cloud.google.com/serverless

   Accessed 21 Oct 2022.

5. IBM Cloud Functions; 2021.

   Available:https://www.ibm.com/cloud/functions.

   Accessed 21 Oct 2022.

6. Alibaba Cloud Function Compute; 2021. Available:https://www.alibabacloud.com/products/function-compute.

   Accessed 21 Oct 2022.

7. AWS Lambda Customer Case Studies; 2021a Available:https://aws.amazon.com/lambda/resources/customer-case-studies/. Accessed 21 Oct 2022.

8. Serverless Computing Market Insights; 2021b. Available:https://www.digitaljournal.com/pr/serverless-computing-market-insights-2022-business-opportunities-current-trends-and-restraints-forecast-2026#ixzz7W67yDNi4.

   Accessed 21 Oct 2022.

9. Hong S, Srivastava A, Shambrook W, Dumitras T. Go Serverless: Securing Cloud via Serverless Design Patterns. In: USENIX Workshop on Hot Topics in Cloud Computing (HotCloud). USENIX Association, Boston; 2018.

10. (2021) OWASP Serverless Top 10. https://owasp.org/www-project-serverless-top-10/. Accessed 21 Oct 2022.

11. AWS Serverless Application Repository; 2021

    Available:https://aws.amazon.com/en/serverless/serverlessrepo/.

    Accessed 21 Oct 2022.

12. Gao X, Gu Z, Li Z, Jamjoom H, Wang C Houdini's Escape: Breaking the Resource Rein of Linux Control Groups. In: ACM SIGSAC Conference on Computer and Communications Security (CCS). Association for Computing Machinery, New York. 2019;1073–1086

13. Nam J, Lee S, Seo H, Porras P, Yegneswaran V, Shin S () BASTION: A Security Enforcement Network Stack for Container Networks. In: USENIX Annual Technical Conference (USENIX ATC). USENIX Association. 2020;81–95

14. Ory Segal: Serverless Security // Serverless Days TLV; 2021a. Available:https://www.youtube.com/watch?v=M7wUanfWs1c &t=743s.

    Accessed 21 Oct 2022.

15. Event Injection: Protecting your Serverless Applications; 2021b. Available:https://www.jeremydaly.com/event-injection-protecting-your-serverless-applications/. Accessed 21 Oct 2022.

16. Yelam A, Subbareddy S, Ganesan K, Savage S, Mirian A. CoResident Evil: Covert Communication In The Cloud With Lambdas. In: the Web Conference (WWW). Association for Computing Machinery, New York. 2021;1005–1016

17. Wang L, Li M, Zhang Y, Ristenpart T, Swift M. Peeking behind the Curtains of Serverless Platforms. In: USENIX Conference on Usenix Annual Technical Conference (USENIX ATC). USENIX Association, Boston. 2018;133–145

18. CVE-2022-0185: Kubernetes Container Escape Using Linux Kernel Exploit; 2022. Available:https://www.crowdstrike.com/blog/cve-2022-0185-kubernetes-container-escape-using-linux-kernel-exploit/.

    Accessed Oct 21 2022.

19. Hacking serverless runtimes: Profiling AWS Lambda, Azure Functions, And more; 2019. Available:https://www.blackhat.com/us-17/briefings/schedule/#hacking-serverless-runtimes-profiling-aws-lambda-azure-functions-and-more-6434.

20. Xiong J, Wei M, Lu Z, Liu Y. Warmonger: Inflicting Denial-of-Service via Serverless Functions in the Cloud. In: ACM SIGSAC Conference on Computer and Communications Security (CCS).. Association for Computing Machinery, New York. 2021;955–969

21. Kelly D, Glavin FG, Barrett E. Denial of wallet–Defining a looming threat to serverless computing. Journal of Information Security and Applications 2021;(60):2214–2126.
22. Many-faced threats to Serverless security;2021.
    Available:https://hackernoon.com/many-faced-threats-to-serverless-security-519e94d19dba.
    Accessed 21 Oct 2022.
23. Liu G, Gao X, Wang H, Sun K. Exploring the Unchartered Space of Container Registry Typosquatting. In: USENIX Security Symposium (USENIX Security). USENIX Association, Boston. 2022;35–51
24. Makrani HM, Sayadi H, Nazari N, Khasawneh KN, Sasan A, Rafatirad S, Homayoun H. Cloak & Co-locate: Adversarial Railroading of Resource Sharing-based Attacks on the Cloud. In: International Symposium on Secure and Private Execution Environment Design (SEED). 2021;1–13.
25. Fang C, Wang H, Nazari N, Omidi B, Sasan A, Khasawneh KN, Rafatirad S, Homayoun H .Repttack: Exploiting Cloud Schedulers to Guide Co-Location Attacks. In: Network and Distributed System Security Symposium (NDSS); 2022
26. Razavi K, Gras B, Bosman E, Preneel B, Giuffrida C, Bos H. Flip Feng Shui: Hammering a Needle in the Software Stack. In: USENIX Security Symposium (USENIX Security). USENIX Association, Austin. 2016;1–18
27. Kocher P, Horn J, Fogh A, Genkin D, Gruss D, Haas W, Hamburg M, Lipp M, Mangard S, Prescher T, Schwarz M, Yarom Y (2019) Spectre Attacks: Exploiting Speculative Execution. In: IEEE Symposium on Security and Privacy (S&P). 2019;1–19.
28. Lipp M, Schwarz M, Gruss D, Prescher T, Haas W, Fogh A, Horn J, Mangard S, Kocher P, Genkin D, Yarom Y, Hamburg M () Meltdown: Reading Kernel Memory from User Space. In: USENIX Security Symposium (USENIX Security). USENIX Association, Baltimore. 2018;973–990.
29. Datta P, Kumar P, Morris T, Grace M, Rahmati A, Bates A. Valve: Securing Function Workflows on Serverless Computing Platforms. In: The Web Conference (WWW). pp. Association for Computing Machinery, New York. 2020;939–950.
30. Sankaran A, Datta P, Bates A. Workflow integration alleviates identity and access management in serverless computing. In: Annual Computer Security Applications Conference (ACSAC). 496–509. Association for Computing Machinery, New York; 2020.
31. Anjali, Caraza-Harter T, Swift MM. Blending Containers and Virtual Machines: A Study of Firecracker and GVisor. In: ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE). Association for Computing Machinery, New York. 2020;101–113
32. Hyper-V Technology Overview; 2021.
    Available:https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview. Accessed 21 Oct 2022.
33. Nabla containers: A new approach to container isolation; 2021
    Available:https://nabla-containers.github.io/.
    Accessed 21 Oct 2022.
34. Kata containers; 2021.
    Available:https://katacontainers.io/.
    Accessed 21 Oct 2022.
35. Ferraiolo DF, Kuhn DR. Role-Based Access Controls;2009. Available:https://doi.org/10.48550/ARXIV.0903.2171.
36. Spiffe: Secure Production Identity Framework for Everyone; 2021.
    Available:https://spiffe.io/.
    Accessed 21 Oct 2022.
37. Corsha: API Identity & Access Management; 2021.
    Available:https://corsha.com/.
    Accessed 21 Oct 2022.
38. The Minimum Elements For a Software Bill of Materials (SBOM);2021. Available:https://www.ntia.doc.gov/report/2021/minimum-elements-software-bill-materials-sbom. Accessed 21 Oct 2022.
39. Gone in 60 Milliseconds: Intrusion and Exfiltration in Serverless Architectures; 2021.
    Available:https://media.ccc.de/v/33c3-7865-gone_in_60_milliseconds.
    Accessed 21 Oct 2022

40. (2021) How AWS Lambda reuses containers (and how it affects you). https://pfisterer.dev/posts/aws-lambda-container-reuse. Accessed 21 Oct 2022.
41. Savi M, Banfi A, Tundo A, Ciavotta M. Serverless Computing for NFV: Is it Worth it? A Performance Comparison Analysis. In: IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops). 2022;680–685.
42. Castro P, Ishakian V, Muthusamy V, Slominski A. The Rise of Serverless Computing. Commun ACM. 2019;62.
43. Colantonio A, Pietro R, Ocello A. Role Mining in Business: Taming Role-Based Access Control Administration. Singapore: World Scientific; 2012
44. Combe T, Martin A, Pietro R. To Docker or Not to Docker: A Security Perspective. IEEE Cloud Comput. 2016;3.
45. Hu VC, Kuhn DR, Ferraiolo DF, Voas J. Attribute-Based Access Control. Computer. 2015;48.
46. Kulkarni SG, Liu G, Ramakrishnan KK, Wood T. Living on the edge: Serverless computing and the cost of failure resiliency In: 2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN). 2019;1–6.
Available:https://doi.org/10.1109/LANMAN.2019.8846970
47. Jambunathan B, Yoganathan K. Architecture decision on using microservices or serverless functions with containers In: 2018 International Conference on Current Trends Towards Converging Technologies (ICCTCT), 2018;1–7.
Available:https://doi.org/10.1109/ICCTCT.2018.8551035.
48. Kuhlenkamp J, Werner S (2018) Benchmarking faas platforms: Call for community participation In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 189–194. https://doi.org/10.1109/UCC-Companion.2018.00055.
49. Somma G, Ayimba C, Casari P, Romano SP, Mancuso V. When less is more: Core-restricted container provisioning for serverless computing In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020;1153–1159. Available:https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162876.
50. Sewak M, Singh S (2018) Winning in the era of serverless computing and function as a service In: 3rd International Conference for Convergence in Technology (I2CT). 2018;1–5. Available:https://doi.org/10.1109/I2CT.2018.8529465.
51. Feng L, Kudva P, Da Silva D, Hu J (2018) Exploring serverless computing for neural network training In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 334–341. https://doi.org/10.1109/CLOUD.2018.00049.
52. Werner S, Kuhlenkamp J, Klems M, Müller J, Tai S. Serverless big data processing using matrix multiplication as example In: 2018 IEEE International Conference on Big Data (Big Data), 2018;358–365.
Available:https://doi.org/10.1109/BigData.2018.8622362
53. Al-Ali Z, Goodarzy S, Hunter E, Ha S, Han R, Keller E, Rozner E (2018) Making serverless computing more serverless In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 456–459. https://doi.org/10.1109/CLOUD.2018.00064.
54. Pérez A, Risco S, Naranjo DM, Caballer M, Moltó G. On-premises serverless computing for event-driven data processing applications In: IEEE 12th International Conference on Cloud Computing (CLOUD). 2019;414–421.
Available:https://doi.org/10.1109/CLOUD.2019.00073.
55. Taibi D, El Ioini N, Pahl C, Niederkofler J. Patterns for Serverless Functions (Function-as-a-Service): A Multivocal Literature Review In: Proceedings of the 10th International Conference on Cloud Computing and Services Science. CLOSER, 2020(1):181–192. Available:https://doi.org/10.5220/0009578501810192.
56. Hellerstein JM, Faleiro J, Gonzalez JE, Schleier-Smith J, Sreekanti V, Tumanov A, Wu C .Serverless Computing: One Step Forward, Two Steps Back; 2018. Available:http://arxiv.org/abs/1812.03651
Accessed 4 Oct 2021.

57. Sadaqat M, Colomo-Palacios R, Knudsen LES. Serverless Computing: A Multivocal Literature Review. NOKOBIT - Norsk Konferanse for Organisasjoners Bruk Av Informasjonsteknologi. 2018;26(1):1–13.
58. Fox GC, Ishakian V, Muthusamy V, Slominski A. Status of Serverless Computing and Function-as-a-Service(FaaS) in Industry and Research. arXiv e-prints. 2017;1708–08028. Available:http://arxiv.org/abs/1708.08028.
    Accessed 6 Jan 2021.
59. Hassan HB, Barakat SA, Sarhan QI. Serverless Literature Dataset. Zenodo; 2021.
    Available:https://doi.org/10.5281/zenodo.4660553
60. Bila N, Dettori P, Kanso A, Watanabe Y, Youssef A. Leveraging the serverless architecture for securing linux containers In: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). 2017;401–404.
    Available:https://doi.org/10.1109/ICDCSW.2017.66
61. Chang KS, Fink SJ. Visualizing serverless cloud application logs for program understanding In: 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2017;261–265.
    Available:https://doi.org/10.1109/VLHCC.2017.8103476
62. Ishakian V, Muthusamy V, Slominski A. Serving deep learning models in a serverless platform In: 2018 IEEE International Conference on Cloud Engineering (IC2E), 2018;257–262. Available:https://doi.org/10.1109/IC2E.2018.00052
63. Parás G, Garcáa-López P, Sánchez-Artigas M (2020) Serverless elastic exploration of unbalanced algorithms In: 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), 149–157.
    Available:https://doi.org/10.1109/CLOUD49709.2020.00033
64. Kuhlenkamp J, Werner S, Tai S. The ifs and buts of less is more: A serverless computing reality check In: 2020 IEEE International Conference on Cloud Engineering (IC2E). 2020;154–161. Available:https://doi.org/10.1109/IC2E48712.2020.00023
65. Pfandzelter T, Bermbach D. Tinyfaas: A lightweight faas platform for edge environments In: 2020 IEEE International Conference on Fog Computing (ICFC). 2020;17–24.
    Available:https://doi.org/10.1109/ICFC49376.2020.00011
66. Bermbach D, Maghsudi S, Hasenburg J, Pfandzelter T. Towards auction-based function placement in serverless fog platforms In: 2020 IEEE International Conference on Fog Computing (ICFC). 2020;25–31.
    Available:https://doi.org/10.1109/ICFC49376.2020.00012
67. Garcia Lopez P, Sanchez-Artigas M, Paris G, Barcelona Pons D, Ruiz Ollobarren A, Arroyo Pinto. Comparison of faas orchestration systems; D2018.
    Available:https://doi.org/10.1109/ucc-companion. 2018.00049.
68. Ichnowski J, Lee W, Murta V, Paradis S, Alterovitz R, Gonzalez JE, Stoica I, Goldberg K. Fog robotics algorithms for distributed motion planning using lambda serverless computing In: 2020 IEEE International Conference on Robotics and Automation (ICRA). 2020;4232–4238. Available:https://doi.org/10.1109/ICRA40945.2020.9196651
69. Gupta V, Carrano D, Yang Y, Shankar V, Courtade T, Ramchandran K. Serverless straggler mitigation using error-correcting codes In: 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), 2020;135–145. Available:https://doi.org/10.1109/ICDCS47774.2020.00019
70. Lloyd W, Ramesh S, Chinthalapati S, Ly L, Pallickara S. Serverless computing: An investigation of factors influencing microservice performance In: 2018 IEEE International Conference on Cloud Engineering (IC2E). 2018;159–169.
    Available:https://doi.org/10.1109/IC2E.2018.00039
71. Lloyd W, Vu M, Zhang B, David O, Leavesley G. Improving application migration to serverless computing platforms: Latency mitigation with keep-alive workloads In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). 2018;195–200.

Available:https://doi.org/10.1109/UCC-Companion.2018.00056.

72. Al-Masri E, Diabate I, Jain R, Lam MHL, Nathala SR. A serverless iot architecture for smart waste management systems In: 2018 IEEE International Conference on Industrial Internet (ICII). 2018;179–180.
Available:https://doi.org/10.1109/ICII.2018.00034

73. Cordingly R, Yu H, Hoang V, Perez D, Foster D, Sadeghi Z, Hatchett R, Lloyd WJ. Implications of programming language selection for serverless data processing pipelines In: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2020;704–711. Available:https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00120.

74. Cordingly R, Shu W, Lloyd WJ (2020) Predicting performance and cost of serverless computing functions with saaf In: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech). 2020;640–649.
Available:https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00111.

75. Toader L, Uta A, Musaafir A, Iosup A. Graphless: Toward serverless graph processing In: 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC). 2019;66–73. Available:https://doi.org/10.1109/ISPDC.2019.00012

76. van Eyk E, Iosup A. Addressing performance challenges in serverless computing In: ICT. OPEN; 2018.

77. Oakes E, Yang L, Houck K, Harter T, Arpaci-Dusseau AC, Arpaci-Dusseau RH. Pipsqueak: Lean lambdas with large libraries In: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 2017;395–400.
Available:https://doi.org/10.1109/ICDCSW.2017.32

78. Aumala G, Boza E, Ortiz-Avilés L, Totoy G, Abad C. Beyond load balancing: Package-aware scheduling for serverless platforms In: 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2019;282–291. Available:https://doi.org/10.1109/CCGRID.2019.00042

79. Manner J, Endreß M, Heckel T, Wirtz G. Cold start influencing factors in function as a service In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). 2018;181–188.
Available:https://doi.org/10.1109/UCC-Companion.2018.00054

80. Winzinger S, Wirtz G. Model-based analysis of serverless applications In: 2019 IEEE/ACM 11th International Workshop on Modelling in Software Engineering (MiSE), 2019;82–88.
Available:https://doi.org/10.1109/MiSE.2019.00020

81. Winzinger S, Wirtz G. Applicability of coverage criteria for serverless applications In: 2020 IEEE International Conference on Service Oriented Systems Engineering (SOSE). 2020;49–56. Available:https://doi.org/10.1109/SOSE49046.2020.00013

82. Gias AU, Casale G (2020) Cocoa: Cold start aware capacity planning for function-as-a-service platforms In: 2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2020;1–8.
Available:https://doi.org/10.1109/MASCOTS50786.2020.9285966

83. Chatley R, Allerton T. Nimbus: Improving the developer experience for serverless applications In: 2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). 2020;85–88.

84. Vandebon J, Coutinho JGF, Luk W, Nurvitadhi E, Naik M Slate: Managing heterogeneous cloud functions In: 2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP). 2020;141–148.

Available:https://doi.org/10.1109/ASAP49362.2020.00032

85. Kim J, Lee K. Functionbench: A suite of workloads for serverless cloud function service In: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD). 2019;502–504. Available:https://doi.org/10.1109/CLOUD.2019.00091

86. Kim J, Park J, Lee K. Network resource isolation in serverless cloud function service In: 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W), 2019;182–187.
Available:https://doi.org/10.1109/FAS-W.2019.00051.

87. Park J, Kim H, Lee K (2020) Evaluating concurrent executions of multiple function-as-a-service runtimes with microvm In: 2020 IEEE 13th International Conference on Cloud Computing (CLOUD).2020;532–536.
Available:https://doi.org/10.1109/CLOUD49709.2020.00080

88. Wu M, Mi Z, Xia Y (2020) A survey on serverless computing and its implications for jointcloud computing In: 2020 IEEE International Conference on Joint Cloud Computing, 94–101. https://doi.org/10.1109/JCC49151.2020.00023.

89. Li Z, Chen Q, Xue S, Ma T, Yang Y, Song Z, Guo M. Amoeba: Qos-awareness and reduced resource usage of microservices with serverless computing In: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2020;399–408. Available:https://doi.org/10.1109/IPDPS47924.2020.00049.

90. Liu J, Mi Z, Huang Z, Hua Z, Xia Y. Hcloud: A serverless platform for jointcloud computing In: 2020 IEEE International Conference on Joint Cloud Computing. 2020;86–93. Available:https://doi.org/10.1109/JCC49151.2020.00022.

91. Wang H, Niu D, Li B. Distributed machine learning with a serverless architecture In: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications. 2019;1288–1296. Available:https://doi.org/10.1109/INFOCOM.2019.8737391

92. Mahmoudi N, Khazaei H. Performance modeling of serverless computing platforms. IEEE Trans Cloud Comput: 2020;1–1.
Available:https://doi.org/10.1109/TCC.2020.3033373

93. Wurster M, Breitenbücher U, Képes K, Leymann F, Yussupov V. Modeling and automated deployment of serverless applications using tosca In: 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA), 2018;73–80. Available:https://doi.org/10.1109/SOCA.2018.00017

94. Yussupov V, Breitenbücher U, Hahn M, Leymann F. Serverless parachutes: Preparing chosen functionalities for exceptional workloads In: 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC). 2019;226–235. Available:https://doi.org/10.1109/EDOC.2019.00035

95. Shahrad M, Fonseca R, Goiri I, Chaudhry G, Batum P, Cooke J, Laureano E, Tresness C, Russinovich M, Bianchini R (2020) Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider In: 2020 USENIX Annual Technical Conference (USENIX ATC 20). 2020;205–218.

96. Das A, Imai S, Patterson S, Wittie MP. Performance optimization for edge-cloud serverless platforms via dynamic task placement In: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID). 2020;41–50. Available:https://doi.org/10.1109/CCGrid49817.2020.00-89

97. Das A, Leaf A, Varela CA, Patterson S (2020) Skedulix: Hybrid cloud scheduling for cost-efficient execution of serverless applications In: 2020 IEEE 13th International Conference on Cloud Computing (CLOUD). 2020;609–618. Available:https://doi.org/10.1109/CLOUD49709.2020.00090.

98. Lin W, Krintz C, Wolski R, Zhang M, Cai X, Li T, Xu W. Tracking causal order in aws lambda applications In: 2018 IEEE International Conference on Cloud Engineering (IC2E). 2018;50–60.

Available:https://doi.org/10.1109/IC2E.2018.00027

99. George G, Bakir F, Wolski R, Krintz C (2020) Nanolambda: Implementing functions as a service at all resource scales for the internet of things In: IEEE/ACM Symposium on Edge Computing (SEC). 2020;220–231.
Available:https://doi.org/10.1109/SEC50012.2020.00035

100. Zhang M, Krintz C, Wolski R (2020) Stoic: Serverless teleoperable hybrid cloud for machine learning applications on edge device In: 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 1–6. https://doi.org/10.1109/PerComWorkshops48775.2020.9156239.

101. Elgamal T. Costless: Optimizing cost of serverless computing through function fusion and placement In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), 2018;300–312. https://doi.org/10.1109/SEC.2018.00029.

102. Malawski M, Gajek A, Zima A, Balis B, Figiela K. Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions. Futur Gener Comput Syst: 2017;1–13.
Available:https://doi.org/10.1016/j.future.2017.10.029.

103. Moczurad P, Malawski M (2018) Visual-textual framework for serverless computation: A luna language approach In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). 2018;169–74.
Available:https://doi.org/10.1109/UCC-Companion. 2018.00052.

104. Akhtar N, Raza A, Ishakian V, Matta I Cose: Configuring serverless functions using statistical learning In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. 2020;129–38.
Available:https://doi.org/10.1109/INFOCOM41043.2020.9155363

105. Balla D, Maliosz M, Simon C (2020) Open source faas performance aspects In: 43rd International Conference on Telecommunications and Signal Processing (TSP). 2020;358–364. Available:https://doi.org/10.1109/TSP49548.2020.9163456

106. Pelle I, Czentye J, Dóka J, Kern A, Gerő BP, Sonkoly B. Operating latency sensitive applications on public serverless edge cloud platforms. IEEE Internet Things J. 2020;1–1.
Available:https://doi.org/10.1109/JIOT.2020.3042428

107. Carver B, Zhang J, Wang A, Cheng Y. In search of a fast and efficient serverless dag engine In: 2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW). 2019;1–10.
Available:https://doi.org/10.1109/PDSW49588.2019.00005

108. Gadepalli PK, Peach G, Cherkasova L, Aitken R, Parmer G. Challenges and opportunities for efficient serverless computing at the edge In: 2019 38th Symposium on Reliable Distributed Systems (SRDS), 2019;261–2615.
Available:https://doi.org/10.1109/SRDS47363.2019.00036

109. Somu N, Daw N, Bellur U, Kulkarni P. Panopticon: A comprehensive benchmarking tool for serverless applications In: 2020 International Conference on COMmunication Systems NETworkS (COMSNETS). 2020;144–151. Available:https://doi.org/10.1109/COMSNETS48256.2020.9027346

110. Kim YK, HoseinyFarahabady MR, Lee YC, Zomaya AY, Jurdak R. Dynamic control of cpu usage in a lambda platform In: 2018 IEEE International Conference on Cluster Computing (CLUSTER). 2018;234–244.
Available:https://doi.org/10.1109/CLUSTER.2018.00041

111. Suresh A, Somashekar G, Varadarajan A, Kakarla VR, Upadhyay H, Gandhi A (2020) Ensure: Efficient scheduling and autonomous resource management in serverless environments In: 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS). 2020;1–10.
Available:https://doi.org/10.1109/ACSOS49614.2020.00020

112. Kim Y, Lin J (2018) Serverless data analytics with flint In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018;451–455. Available:https://doi.org/10.1109/CLOUD.2018.00063

113. Parres-Peredo A, Piza-Davila I, Cervantes F (2019) Building and evaluating user

network profiles for cybersecurity using serverless architecture In: 2019 42nd International Conference on Telecommunications and Signal Processing (TSP). 2019;164–167.

Available:https://doi.org/10.1109/TSP.2019.8768825

114. Deese A. Implementation of unsupervised k-means clustering algorithm within amazon web services lambda In: 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). 2018; 626–632.

Available:https://doi.org/10.1109/CCGRID.2018.00093

115. Kritikos K, Skrzypek P. A review of serverless frameworks In: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). 2018;161–168.

Available:https://doi.org/10.1109/UCC-Companion.2018.00051

116. Kritikos K, Skrzypek P (2019) Simulation-as-a-service with serverless computing In: 2019 IEEE World Congress on Services (SERVICES). 2019;2642-939X:200–205. Available:https://doi.org/10.1109/SERVICES.2019.00056

117. Christoforou A, Andreou AS. An effective resource management approach in a faas environment In: ESSCA@UCC. 2018;2–8.

© 2023 Cinar; This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle5.com/review-history/107654