Volume: 13 Issue: VII Month of publication: July 2025

DOI: https://doi.org/10.22214/ijraset.2025.73159

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VII July 2025- Available at www.ijraset.com

Micro Service-Based, Containerized, & Server Less Applications in Cloud

Rupinder Kaur[1], Roopi[2], Mandeep Kaur[3]
Anand college of Engineering and Management, KPT

Abstract: The use of cloud-based software solutions has become increasingly popular in recent years. As a result, the use of microservices-based, containerized, and server less architectures for the deployment, management, and scaling of cloud-based applications has grown in popularity. This review paper provides an overview of containerized, server less architectures, and microservices in cloud-based applications. It demonstrates how containers provide a lightweight and portable approach to package apps and their dependencies, as well as how microservices involve dividing an application into smaller, independent services. Server less computing is also discussed, which enables programmers to create and execute applications without managing infrastructure. The benefits and limitations of server less, containers, and micro service architectures in cloud based applications are covered in this research. It explains how combining these technologies can result in adaptable, scalable, fault-tolerant applications that also solve security issues. The paper underlines the significance of developing scalable applications from the outset and offers best practises for protecting and monitoring these distributed architectures. The article also looks at new trends and technologies that could influence how these technologies are used in cloud-based applications in the future. Overall, it provides a thorough review of these technologies' potential for creating effective and potent cloud-based applications.

Keywords: Microservices, Containerized, Server less, Deploy, Benefits and Challenges, Cloud Computing, Cloud Applications

I. INTRODUCTION

Cloud computing has transformed the way applications are built and managed. This shift has been quickened by the emergence of microservices, containers, and server less computing. A method for creating software systems that is made up of small, autonomous services that communicate with one another using APIs is called microservices architecture. Using containers to deploy and operate apps in various contexts is a portable and lightweight approach. This review paper aims to provide an overview of micro service-based, containerized, and server less applications in the cloud. We will discuss the benefits, challenges, and best practices of these approaches. We will also review some of the popular platforms and tools that are used to build, and manage these applications. Micro service-based architecture is a method of designing applications as a collection of independent, modular services that communicate directly with one another to create a larger application. Because each micro service can be written, tested, and delivered independently, it enables enterprises to design and deploy software applications more quickly. Each service is intended to serve a specific purpose and communicate with other services via well-defined interfaces. Also, this

Strategy makes it simpler to scale individual services in accordance with demand, which enhances application performance and lowers costs. Microservices do, however, also bring with them new difficulties, such as a more difficult time monitoring and coordinating the various services. Containers are fundamental to the deployment and development of modern applications. The procedure of packaging an application and its dependencies into a single, portable unit is known as containerization. For packaging and deploying applications across various settings, such as on premises data centres, public clouds, and hybrid clouds, they offer a compact and transportable solution. Bin (Containers), in any case of the underlying infrastructure, provide a consistent and valid environment in which applications can run. Containers also give developers a consistent management of containerized systems may be difficult, especially when it comes to orchestration and networking. A new development in cloud computing is server less computing. Developers can concentrate on building code rather than maintaining the supporting infrastructure because of it. Applications are divided up into smaller functions in a server less architecture, and these smaller functions are then executed in response to events. Since developers only pay for the resources, they use when their functions are actually performed, this method can shorten development time and cost. But server less also brings with it some new difficulties, like potential cold-start latency, vendor lock-in, and restricted control over the infrastructure. Several platforms and tools, including Amazon Lambda, Azure Functions, and Google Cloud Functions, have been developed to tackle these problems. These three architectures are the future of cloud computing. They offer more flexibility, scalability, and agility than conventional monolithic applications, among other advantages. But they also bring about fresh issues that must be resolved. To succeed, organisations using these approaches must carefully examine their needs, select the best platforms and tools, and adopt best practices. This review paper will give a thorough overview of these strategies and the key concepts, benefits, challenges, and best practices of micro service-based, containerized, and server less applications in the cloud environment in which to operate, increasing efficiency and lowering the possibility of compatibility problems. Yet, scaling up the management of containerized systems may be difficult, especially when it comes to orchestration and networking. A new development in cloud computing is server less computing. Developers can concentrate on building code rather than maintaining the supporting infrastructure because of it. Applications are divided up into smaller functions in a server less architecture, and these smaller functions are then executed in response to events. Since developers only pay for the resources, they use when their functions are actually performed, this method can shorten development time and cost. But server less also brings with it some new difficulties, like potential cold-start latency, vendor lock-in, and restricted control over the infrastructure. Several platforms and tools, including Amazon Lambda, Azure Functions, and Google Cloud Functions, have been developed to tackle these problems. These three architectures are the future of cloud computing.

II. BACKGROUND

In past few years, there has been a significant shift towards cloud computing as a means of delivering software applications and services as traditional monolithic applications that are deployed on the cloud is difficult to manage and scale. Cloud computing provides several benefits over traditional on-premises computing, including scalability, flexibility, and cost-effectiveness. As a result, several companies are adopting cloud computing and developing services and applications that are optimised for the cloud.

A. Microservices-based Applications

Applications built using the Microservices architectural pattern consist of several smaller, autonomous services that interact with one another via APIs. Each service is capable of independent development, deployment, and scaling while carrying out a particular business function. Developers can create complex applications using the microservices architecture by decomposing large monolithic apps into smaller services that can be independently created and delivered. Applications built using microservices provide a number of advantages, including scalability, agility, and resilience.

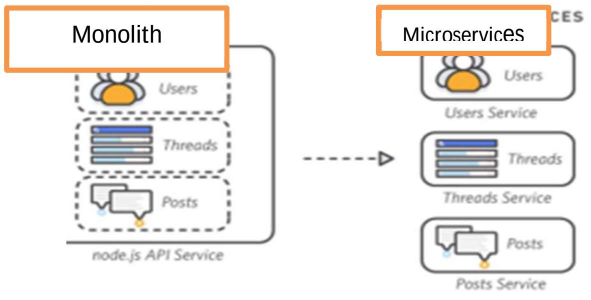


Fig. 1 Monolith vs Microservices

1) Scalability: The capacity to scale services horizontally using microservices architecture allows them to manage a range of workloads. By replicating services and dividing the workload among several instances, this is accomplished.
2) Agility: Because microservices architecture allows for quick and frequent service deployment, organisations are better equipped to react swiftly to shifting business requirements. This is accomplished by dividing monolithic programmes into more manageable services that can be created and deployed individually.
3) Resilience: The architecture of microservices allows for fault isolation and recovery, guaranteeing that if one service fails, it does not affect the entire application. This is accomplished by deploying efficient monitoring and recovery mechanisms and designing services to be fault tolerant.

B. Containerized Applications

A programme and its dependencies can be packaged using the containerization technique into a container image that can be installed and used on any platform that supports containers. Applications can execute in a portable, lightweight environment thanks to containers. By hiding the underlying infrastructure, containerization enables developers to design and deploy apps more quickly and consistently. Containers have a variety of advantages, including portability, isolation, and consistency. Containers are extremely portable since they may be set up on any platform that accepts them.

Server less-based Applications The cloud service provider administers the infrastructure and automatically grows the resources based on the demands of the application in a server less computing environment. Developers create and deploy server less calculate functions that are triggered by events like HTTP requests or database modification. Server less applications offer a few advantages, including affordability, scalability, and flexibility.

Cost-effectiveness: Server less computing reduces the cost of running applications by removing the requirement to deploy and manage infrastructure. This is accomplished by basing charges on the quantity of requests and length of function execution.

Scalability: Server less computing automatically scales resources in accordance with the requirements of the application, ensuring that the programme can handle a range of workloads. To do this, resources are automatically scaled and provisioned based on the volume of requests and the length of the function execution.

III. DEPLOYMENT AND MANAGEMENT

Microservices-based deployment strategy: As part of a microservices-based deployment strategy, a monolithic application is broken up into smaller, independent services that can be deployed and scaled independently. all service execute and determined function and commune with other services using APIs. Increased scalability, fault tolerance, and agility are just a few benefits of micro services based deployment in the cloud. Decoupling the services allows organizations to adjust services without affecting the overall application. T Organizations generally utilise orchestration tools like Kubernetes and containerization technologies like Dockers to deploy microservices in the cloud. To ensure that services are distributed across several nodes for greater fault-tolerance and availability, they can install and manage microservices at scale thanks to this.

Containerized Deployment Strategy: Containerized deployment is a popular method for deploying software in the cloud. In this method, an application is packaged in a container that also houses all its dependencies and configuration settings. Because of the consistent and portable environment provided by containers, applications may be simply deployed and managed across different cloud environments. The containerized deployment method has many benefits, including more agility and scalability, better resource management, and increased application reliability. With containers, applications can be easily updated and deployed, and new instances can be quickly produced to meet changing demand.

To adopt a containerized deployment strategy, developers use tools like Dockers and Kubernetes to construct and manage containers. modern cloud services due to its benefits in terms of scalability, efficiency, and deployment simplicity.

A. Server less-based Deployment Strategy

The cloud service provider administers the infrastructure and automatically grows the resources based on the demands of the application in a server less computing environment. As a result, programmers can create and publish functions that are activated by various kinds of events, such HTTP requests or database modifications. Server less apps can be deployed in the cloud using a variety of server less-based deployment techniques. Using Function-as-a-Service (FaaS) platforms, such as AWS Lambda or Google Cloud Functions, which offer a straightforward programming model for creating and delivering functions, is one popular tactic.

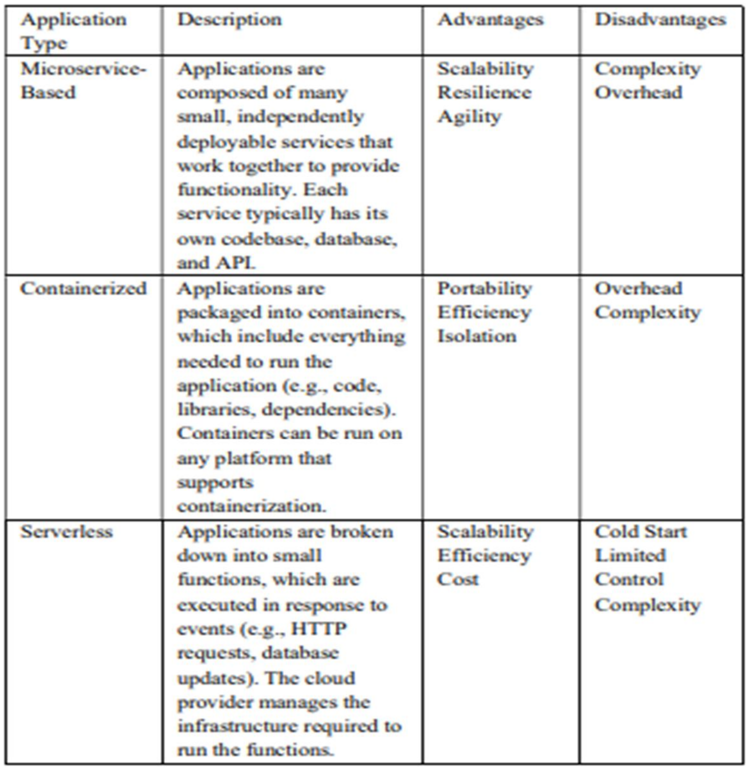
Using API Gateway, which enables programmers to create APIs that can launch server less functions, is another tactic. Moreover, developers can utilise platforms that offer containers as a service (CaaS), such as AWS Far gate or Google Kubernetes Engine, to deploy server less applications in containers.

The advantages of server less-based deployment methodologies include flexibility, scalability, and cost-effectiveness. They increase developer productivity by allowing them to concentrate on writing code rather than worrying about maintaining infrastructure.

IV. CHALLENGES AND LIMITATIONS

These architectures offer several benefits, including scalability, flexibility, and cost-efficiency. However, there are also challenges and limitations associated with these approaches that must be addressed to ensure their success. Complexity and management overhead Microservices can increase the complexity of applications, making it more challenging to manage, test, and deploy. Containerization and server less architectures can add to this complexity, requiring additional tools, frameworks, and expertise. Moreover, managing many containers or server less functions can be overwhelming, and it can be difficult to monitor and troubleshoot issues.

1) Security: Microservices, containers, and server less functions can increase the attack surface of an application, making it more vulnerable to security threats. Each micro service, container, or function must be secured independently, and it can be challenging to ensure that all dependencies are properly secured. Additionally, server less functions can be susceptible to injection attacks and other vulnerabilities, and it can be challenging to secure and monitor them.
2) Performance: While microservices, containers, and server less functions offer scalability, they can also introduce performance issues. Containers and server less functions have limitations on the number of resources available, which can lead to performance degradation if not managed correctly. Additionally, the use of multiple services can introduce network latency and communication overhead, impacting overall application performance.
3) Dependency Management: Microservices, containers, and server less functions rely on a complex web of dependencies, making it challenging to manage updates and changes. Upgrading one micro service or container can have a cascading effect on other services, and it can be challenging to ensure that all dependencies are updated and tested properly. Similarly, server less functions often rely on external libraries and frameworks, making it difficult to manage and test dependencies
4) Debugging and troubleshooting: With the distributed nature of microservices, containerized, and server less applications, it can be challenging to identify and troubleshoot issues. Logs and monitoring tools must be used to track issues across multiple services, making it more difficult to identify the root cause of problems. Similarly, debugging server less functions can be challenging since they are often run in a stateless environment and may not provide access to the underlying infrastructure.
5) Vendor lock-in: Finally, while cloud providers offer powerful tools and services for microservices, containerized, and server less applications, there is a risk of vendor lock-in. Developers must be careful when choosing cloud providers and technologies, as changing providers or moving applications to an on-premises environment can be challenging and time consuming.



Application Type	Description	Advantages	Disadvantages
Microservice-Based	Applications are composed of many small, independently deployable services that work together to provide functionality. Each service typically has its own codebase, database, and API.	Scalability Resilience Agility	Complexity Overhead
Containerized	Applications are packaged into containers, which include everything needed to run the application (e.g., code, libraries, dependencies). Containers can be run on any platform that supports containerization.	Portability Efficiency Isolation	Overhead Complexity
Serverless	Applications are broken down into small functions, which are executed in response to events (e.g., HTTP requests, database updates). The cloud provider manages the infrastructure required to run the functions.	Scalability Efficiency Cost	Cold Start Limited Control Complexity

Fig. 2 Comparison between the advantages and disadvantages of microservice-based, containerized and serverless applications in cloud.

V. BEST PRACTICES

Micro service-based, containerized, and server less architectures offer several benefits for cloud-based applications, including scalability, flexibility, and cost efficiency. However, to fully realize the benefits of these approaches, developers must follow best practices that ensure the reliability, security, and maintainability of the application.

Design with a focus on business capabilities when designing microservices, developers should focus on business capabilities rather than technical capabilities. This means that each micro service should be responsible for a specific business capability or functionality. This approach helps ensure that the microservices are aligned with the business requirements, making the application more flexible and easier to maintain. Use lightweight and standardized communication protocols. Micro services rely on communication protocols to interact with each other. Developers should use lightweight and standardized protocols such as HTTP, REST, or GRPC to ensure that the services can communicate effectively. Additionally, developers should design APIs with a clear and consistent interface to reduce the risk of errors and miscommunication.

1) Implement containerization best practices. When using containers, developers should follow best practices to ensure the reliability and security of the application. This includes minimizing the size of the container, implementing security measures such as image scanning and secrets management, and using container orchestration tools like Kubernetes or Docker Swarm.
2) Optimize server less function performance. When using server less functions, developers should optimize the performance of the function by minimizing cold starts and optimizing resource usage. This includes choosing the right trigger, reducing the size of the function package, and implementing caching and connection pooling techniques.
3) Automate testing and deployment. Automated testing and deployment are critical to ensuring the reliability and scalability of microservices, containerized, and serverless applications. Developers should implement automated testing at every stage of the development cycle, including unit tests, integration tests, and end-to-end tests. Additionally, developers should use continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment of the application.
4) Monitor and troubleshoot issues proactively: With the distributed nature of microservices, containerized, and server less applications, it is essential to monitor and troubleshoot issues proactively. Developers should use monitoring tools to track performance metrics, error rates, and other critical indicators. Additionally, developers should implement centralized logging and tracing to help identify and troubleshoot issues across multiple services.
5) Use cloud provider-agnostic architectures. To avoid vendor lock-in, developers should use cloud provider-agnostic architectures that can run on multiple cloud platforms. This includes using open-source tools and technologies, avoiding cloud-specific APIs, and using container orchestration tools that can run on multiple cloud providers.
6) Secure the application at every level. Finally, security should be a top priority when designing microservices, containerized, and server less applications. Developers should

Implement security measures such as identity and access management, encryption, and network security. Additionally, developers should use security-focused design patterns such as the strangler pattern to gradually migrate the application to a more secure architecture.

VI. FUTURE DIRECTIONS

Micro services, containers, and server less computing have revolutionized the way we develop, deploy, and operate applications in the cloud. These technologies have enabled organizations to build scalable and resilient applications that can quickly adapt to changing business needs. In this article, we will discuss the future directions of Increased adoption of Kubernetes: Cabernets has become the de-facto standard for container orchestration in the cloud. It produce a powerful platform for scaling and managing containerized applications. In the future, we can expect Kubernetes to become even more ubiquitous, with more organizations adopting it to manage their containerized workloads.

Rise of server less architectures: Server less computing has gained a lot of popularity in recent years. In the future, we can expect server less architectures to become even more prevalent, with more organizations adopting them for their applications. Continued growth of micro services, Micro services have become a popular architectural style for building cloud-native applications. They allow institution to break down complex applications into smaller, independent services that can be developed, deploy, and scale independently. In the future, we can expect microservices to continue to grow in popularity, with more organizations adopting them for their applications. Increased use of AI and Machine Learning: Artificial intelligence (AI) and machine learning (ML) have become an integral part of many applications. In the future, we can expect these technologies to become even more prevalent in micro service-based, containerized, and server less applications in the cloud. This will enable organizations to build smarter applications that can learn from data and improve over time. More emphasis on observability: Observability is the ability to understand how an application is behaving in production. It has become increasingly important in micro service-based, containerized, and server less applications, where there are many moving parts. In the future, we can expect more emphasis on observability, with organizations investing in tools and processes to gain insights into how their applications are performing in production.

In the future, we can expect more organizations to adopt these strategies, with micro service-based, containerized, and server less applications playing a key role in enabling seamless application portability across different cloud environments. Greater emphasis on security: Security is a critical concern for any application running in the cloud. In the future, we can expect greater emphasis on security, with organizations investing in tools and processes to ensure the security of their micro service-based, containerized, and server less applications.

VII. CONCLUSION

In conclusion, micro service-based, containerized, and server less applications have transformed the way we build, deploy, and operate applications in the cloud. These technologies provide organizations with the ability to build scalable, resilient, and flexible applications that can quickly adapt to changing business needs. Kubernetes has become the de-facto standard for container orchestration in the cloud, and we can expect its adoption to continue to rise in the future. Server less computing has gained a lot of popularity in recent years, and we can expect server less architectures to become even more prevalent in the future. Microservices have become a popular architectural style for building cloud-native applications, and we can expect their adoption to continue to grow.

As AI and machine learning become more prevalent, we can expect them to play a key role in micro service-based, containerized, and server less applications. Observability has become increasingly important in these applications, and we can expect more emphasis on observability in the future. Hybrid and multi-cloud strategies have become increasingly popular, and we can expect micro service-based, containerized, and server less applications to play a key role in enabling seamless application portability across different cloud environments. Finally, security is a critical concern for any application running in the cloud. We can expect greater emphasis on security, with organizations investing in tools and processes to ensure the security of their micro service-based, containerized, and server less applications. In summary, micro service-based, containerized, and server less application are here to stay, and they will continue to transform the way we build, deploy, and operate applications in the cloud. Organizations that embrace these technologies and invest in the necessary tools and processes will be well-positioned to take advantage of the benefits they provide.

REFERENCES

[1] Lee, H., Satyam, K., and Fox, G. (2018). Evaluation of production server less computing environments. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pages 442–450.
[2] Lin, W.-T., Krintz, C., Wolski, R., Zhang, M., Cai, X., Li, T., and Xu, W. (2018). Tracking causal order in aws lambda applications. 2018 IEEE International Conference on Cloud Engineering (IC2E), pages 50–60.
[3] Lloyd, W., Ramesh, S., Chinthalapati, S., Ly, L., and Pallickara, S. (2018). Serverless computing: An investigation of factors influencing microservice performance. In 2018 IEEE International Conference on Cloud Engineering (IC2E), pages 159–169.
[4] Lynn, T., Rosati, P., Lejeune, A., and Emeakaroha, V. (2017). A preliminary review of enterprise serverless cloud computing (function-as-aservice) platforms. In 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pages 162–169.
[5] Mazlami, G., Cito, J., and Leitner, P. (2017). Extraction of micro services from monolithic software architectures. In 2017 IEEE International Conference on Web Services (ICWS), pages 524–531.
[6] Mohan, A., Sane, H., Doshi, K., Edupuganti, S., Nayak, N., and Sukhomlinov, V. (2019). Agile cold starts for scalable server less. In Proceedings of the 11th USENIX Conference on Hot Topics in Cloud Computing, HotCloud'19, page 21, USA. USENIX Association.