ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 4, pp: (33-37), Month: October - December 2022, Available at: www.researchpublish.com

Containerized-monolithic applications are different from microservices

Aqeel Alkhawaja

Computer Operations Department / Saudi Aramco

DOI: https://doi.org/10.5281/zenodo.7261068

Published Date: 28-October-2022

Abstract: **Monolithic applications have been developed for decades, satisfying market needs. Microservices add value for their service reusability, despite their long deployment process. But some legacy application developers have misunderstood the concept of microservices. Exposing an API service to other applications does not fulfill the microservices criteria. There are other factors such as the hosting platform and deployment method. Developers will normally need to refactor their applications to build structured microservices. Additionally, deploying a monolithic application using container technology, does not transform monolithic applications into microservices as containers.**

Keywords: **Monolithic applications, Microservices, API service.**

1. INTRODUCTION

Application development and deployment are at the core of the IT industry. There has been significant investment in this area to enable businesses, and meet their on-demand requests and needs. Due to this investment, there are multiple options in the market catering to every developer's needs to build desired applications, and agile delivery. Monolithic application development and standard web hosting, or application hosting platforms, is the traditional approach to develop and deploy applications. In recent years, the concept of microservices and containers became popular as they interacted with cloud-native computing solutions (Basyildiz, 2019). With that, businesses have moved to transform their applications into containerized solutions, and started calling their applications microservices. I believe that there is confusion between the concept of containerized monolithic application, and microservices, as these two are completely different. In this article, we will define monolithic applications and microservices, and how containers serve both development approaches.

1.1 Monolithic application development:

What is a monolithic application? It is defined as a combination of software that performs business logic, associated graphical user interfaces, and connections to database or databases, into a single package that runs on a specific platform (Harris, n.d.). This indicates strong dependencies between different application functions, and modules (Figure 1).

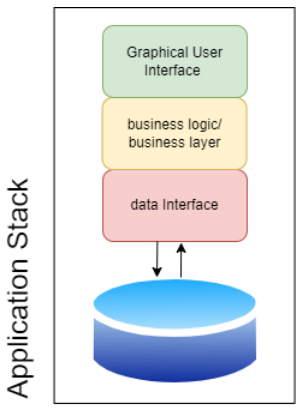


Figure 1

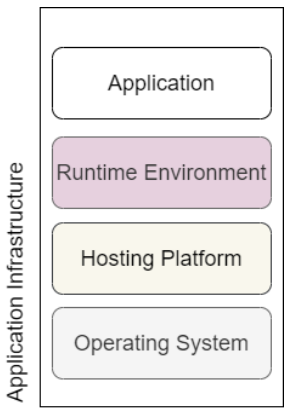


Figure 2

Figure 1 shows a monolithic application's components stack, and their dependencies. The application interface is tightly integrated with business logic, alongside data manipulation components. This is the monolithic approach, a single package that has all applications components included. The application package then gets deployed on a specific platform that fulfills the runtime requirements, and meets operations system dependencies (Figure 2).

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 4, pp: (33-37), Month: October - December 2022, Available at: www.researchpublish.com

1.2 Microservices application development:

A microservices-based application is completely different from a monolithic application. To conduct an analogy, we have to define what a microservice-based application is. It is simply an application architecture where a monolithic application is broken into smaller units. Each unit is considered a tiny application that has its own function and database and runs independently of other units (IBM, 2021) (Figure 3). Each microservice unit is accessible through an application programming interface (API) endpoint, which allows units to exchange data, and invoke specific modules (Figure 4).

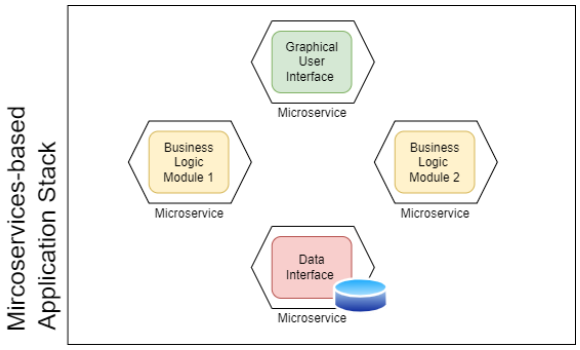


Figure 3

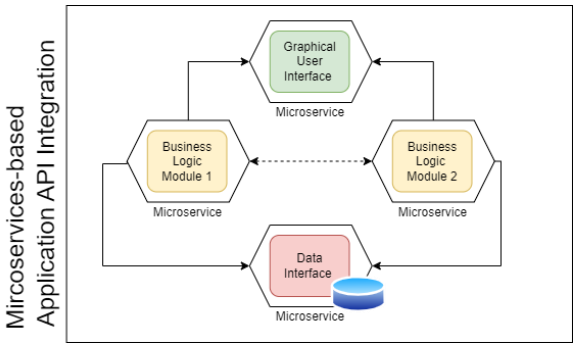


Figure 4

The mentioned definition is from an architectural point of view. From a coding prospective, the proper syntax for each microservice is to handle a single function, then expose it to other components to invoke. In the end, the purpose of the microservices is chunking the monolithic application into pieces. Personally, I think microservices are complex to architects in the beginning, but simplifies the maintenance in the long-term, due to its independency and rapid deployment.

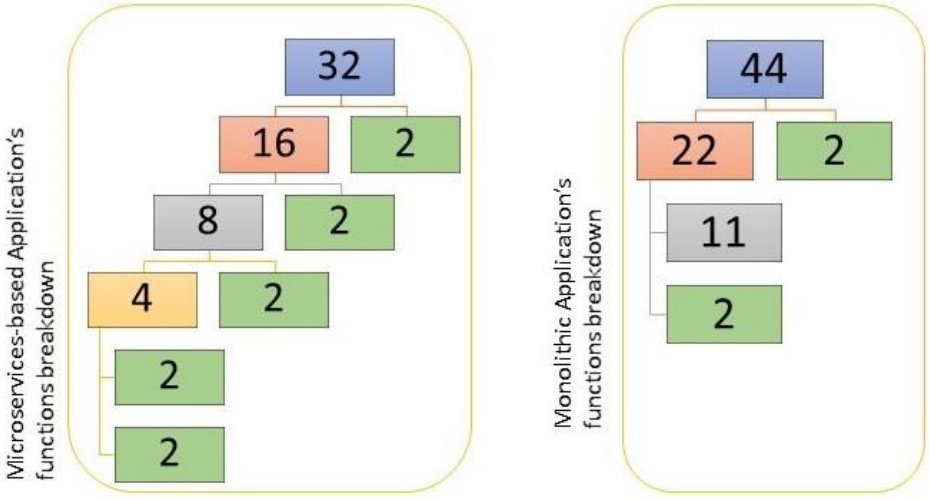
2. ANALOGY BETWEEN MICROSERVICES AND MONOLITHIC APPLICATIONS:

From architecture, and syntax perspective, we can identify major differences between the two development approaches. Now, what are we gaining by following either of the two? Which method will provide a solid application architecture, and optimized performance? There is no right or wrong answer as multiple factors play a role in deciding the path to follow. Let us compare in two different ways: application development level, and deployment level (Harris, n.d.).

	Monolithic application	**Microservices**
Varieties of Technologies	Constrained by single technology.	Flexible to utilize one or more technologies.
Application Error	Single point of failure as the whole application is affected.	Only the impacted microservice will be broken.
Technology Stack	An update on the technology stack could result in recoding the entire application.	An update will impact the single microservice that holds one function of the application.
Deployment	Easy deployment, as it is a single package.	Complex deployment with multiple packages.
Supportability	Changing supporting entities of the application is difficult.	Easily transferred between entities.
Reusability	Difficult to integrate with other applications.	Easy to integrate with other applications.

These are just a few samples from a long running list of differences between the two methods. Major differences lead to different impacts. If I am representing monolithic applications and microservices mathematically, it will be like this:

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 4, pp: (33-37), Month: October - December 2022, Available at: www.researchpublish.com



From the mathematical example, it is clear that microservices can be broken into the smallest units possible, while monolithic will still have major components that cannot be broken.

Up to now, containers are not in the picture, nor are making difference in the way monolithic applications and microservices are deployed. So, what are containers? How are they different from standard hosting platforms? How do these deployment methods impact the monolithic application and microservices?

3. CONTAINERS, AND STANDARD HOSTING PLATFORMS:

What are containers? What are standard platforms? An application developer has to know the differences between the two environments to decide which development path to follow.

Firstly, we have to define what a container is. A container is a computing unit that provides both CPU and memory (IBM, 2021). Containers are different from servers or virtual machines since the container concept is just computing power, but regular servers and virtual machines are a complete operating system stack. For an application to be deployed as containerized solution, it should follow the development standard provided by OCI, Open Container Initiative, to create a container image that can run on different container platforms. Now, we have not talked about a container image, or mentioned what it is. Simply, a container image is wrapping an application source code with required operating system libraries into a single template that can run anywhere using containers (Kubernetes, n.d.). Hence, each container image has the application dependencies bundled within, and isolated from other applications.

Standard hosting platforms are the default approach to deploy applications. Meaning, a single server, that has an operating system, then a platform that handles the runtime requirements for the application. Figure 5 is an overview of IT infrastructure topology of the standard hosting platforms and containers platform.

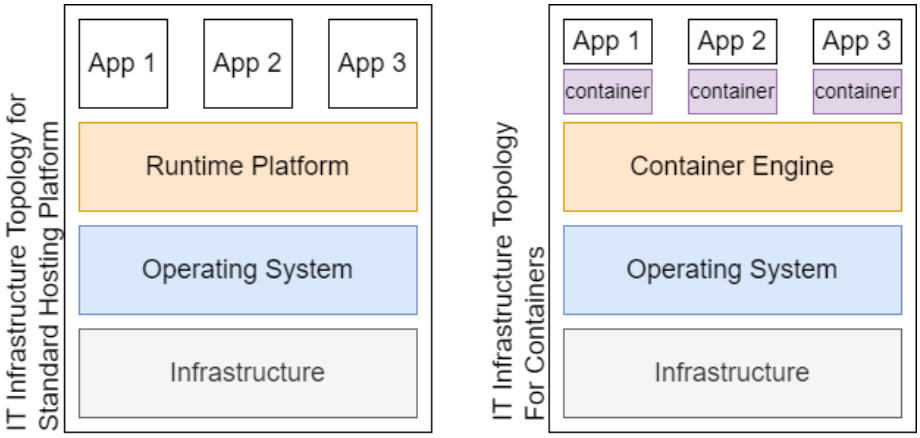


Figure 5

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 4, pp: (33-37), Month: October - December 2022, Available at: www.researchpublish.com

4. WHICH APPROACH FIT WHICH TOPOLOGY

After defining the two different development approaches, and identifying the differences between IT infrastructure topologies for standard hosting platforms and containers, the question that remains to be answered is: what goes with what? The answer is simple. There is no standard regarding which application development approach should go with a specific IT infrastructure. The question should be what is the best fit for each approach? In my opinion, both development approaches can fit containers and standard platforms. It is more into the application requirements and use cases that will determine the proper hosting technology. Monolithic applications are not usually architected to expose public APIs, or services as there are likely no public calls that can be made. Unlike microservices, it aims to expose most of its functions to public through APIs for reusability. I tend to prefer deploying microservices on containers as it gives the freedom of using multiple programming languages, with standard integration with other components using APIs. It also enables rapid deployment since small units are quickly replaced, not the entire application. I would argue monolithic might not be the best approach to containerized, but can still make a noticeable impact on the application, especially from a scalability point of view.

5. CONTAINERIZED APPLICATIONS ARE NOT MICROSERVICES

After going through the definition of monolithic application, microservices, containers, and standard hosting platforms. It is clear that containerized applications do not equal microservices. Containers are just one way of many deployment options. It does not touch an application's structure, it impacts an application's infrastructure. Monolithic can be broken into smaller pieces and have some public APIs exposed, but that is half-way refactoring. It remains a monolithic application with some microservices exposed.

6. EXAMPLE OF A USE

Let us consider a small business application that does currency conversion. A standard monolithic structure will look like Figure 6. While microservices approach will be similar to Figure 7.

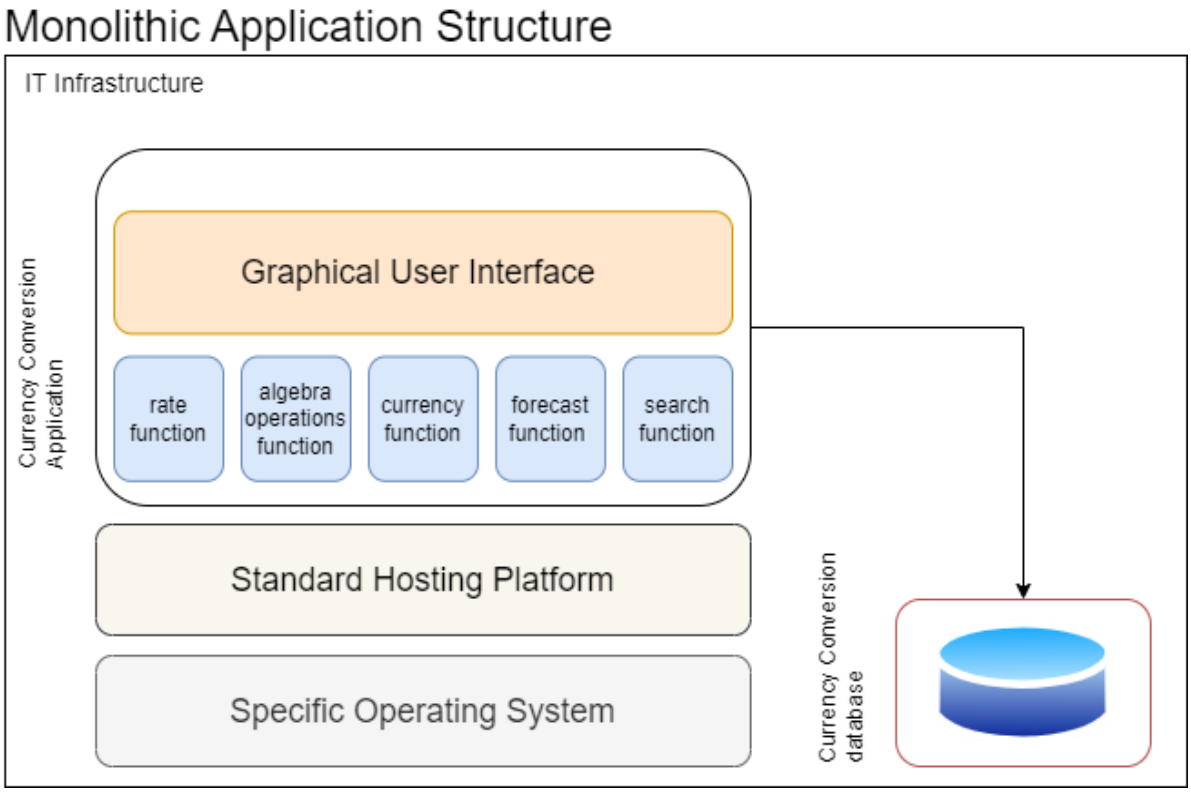


Figure 6

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 10, Issue 4, pp: (33-37), Month: October - December 2022, Available at: www.researchpublish.com

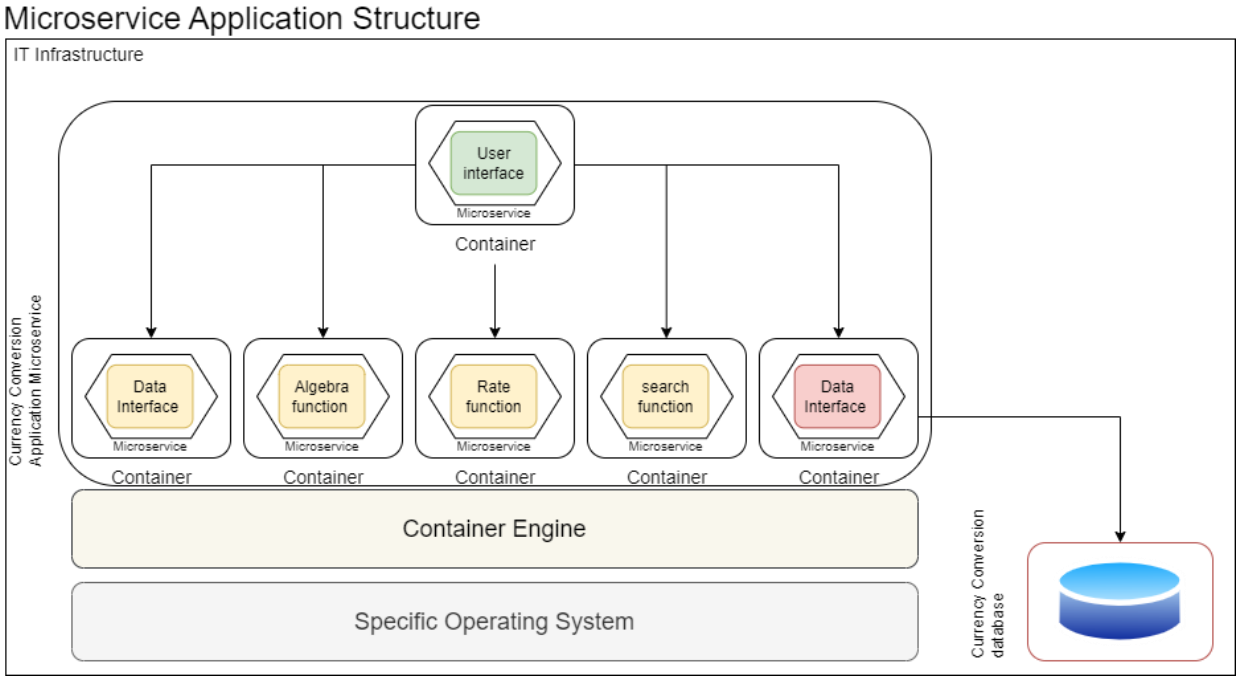


Figure 7

From Figure 6, we can see the tight integration between the user interface, and other functions within the application. From this structure, fitting the application inside a container, or stand hosting platform will not have noticeable performance differences. Monolithic application development indicates high dependency between code functions. If one of the functions is exposed as an API microservice, and the application breaks for any single change on any of the layers, both application and microservice will be unavailable.

Unlike the architecture on Figure 7, each microservice function is running on its own container, and communications are through API calls. This is why microservices are the best fit for containers. A change in one of the microservice will require a reload of that microservice, not the entire application stack. A microservice can be exported to other applications for reusability and it is isolated from the rest of the application, and the availability of other microservices does not impact the rest.

For that reason, application developers have to refactor their code to transform the application architecture from monolithic to microservices approach where suitable to better utilize containers technology and capitalize on its features.

7. SUMMARY

Containerized monolithic applications are not microservices. The development approaches used are different in the application architecture. It is no doubt that monolithic applications can benefit from containers features such as scalability, but they are still considered monolithic. Microservices are meant for service reusability and rapid deployment without impacting the overall service.

REFERENCES

[1] Basyildiz, B. (2019, August 19). A Brief History of Container Technology. Section. https://www.section.io/engineering-education/history-of-container-technology/

[2] Harris, C. (n.d.). Microservices vs. monolithic architecture. Atlassian. https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith#:~:text=A%20monolithic%20application%20is%20built,of%20smaller%2C%20independently%20deployable%20services.

[3] IBM. (2021, June 23). Containerization. IBM. https://www.ibm.com/cloud/learn/containerization

[4] Kubernetes. (n.d.). Images. Kubernetes. https://kubernetes.io/docs/concepts/containers/images/