**MICROSERVICES-Software Development Architecture**

**By Ain Ul Haq**

* **INTRODUCTION:**

Microservices are a software development technique —a variant of the service-oriented architecture (SOA) structural style— that arranges an application as a collection of loosely coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight.

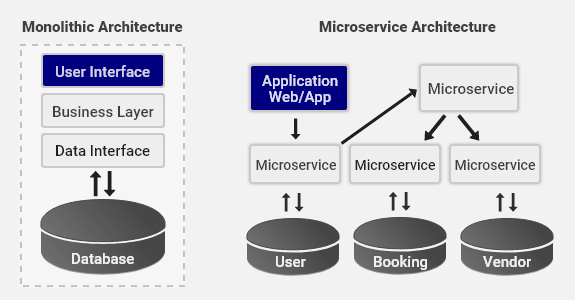
Although many others Monolithic architectures are in use as well, we first differentiate in both the architectures then we will get to know why microservices are important and most useable now a days.

* **DIFFERENCE BETWEEN MONOLITHIC AND MICROSERVICES ARCHITECTURE:**

A monolithic architecture is built as one large system and is usually one code-base. A monolith is often deployed all at once, both front and end code together, regardless of what was changed.

 A microservices architecture however is where an app is built as a suite of small services, each with their own code-base. These services are built around specific capabilities and are usually independently deployable.

In the Below given diagram we can see that in the monolithic architecture every aspect is dependent on another aspect where as in microservices architecture every aspect is independent of the other.

****

* **WHY MICROSERVICES:**

A microservices approach encapsulates each business capability into individual services. Every application process functions as a separate, loosely coupled service with its own logic and database. Updating, deployment, testing and scaling happens within the scope of each service. While microservices don’t reduce the complexity of a system, they do make complexity more visible and manageable. Depending on need, the same service can be reused in several business processes and over a myriad of channels and touchpoints. By standardizing on contracts via business-oriented APIs, users don’t notice any changes made in the backend.

So It’s better to use microservices architecture these days.

* **KEY BENEFITS OF MICROSERVICES:**

Key Benefits of microservices architecture are as follows:

* **Isolation:**

Microservices are profitable due to their isolation and resilience.  If one of the components fail, developers have the option to use another service and the application will continue to run independently. This way, engineers can build and deploy services without the need to change the whole app.

* **Scalability:**

With the architecture of microservices based on small components, it’s easier for development teams to scale up or down following the requirements of a specific element. Isolation allows apps to run correctly when massive changes are happening. Microservices prove to be the perfect approach for companies working with various platforms and devices.

* **Productivity:**

Part of functionality associated with microservice architecture is the ability to easily understand when compared to an entire monolithic app. If you plan to expand your development team, microservices are a better pick.

* **Flexibility:**

The microservice approach lets developers choose the right tools for the right task. They can build each server utilizing a language or framework they need without affecting the [communication between microservices.](https://stackify.com/communication-microservices-avoid-common-problems/)

* **Faster project development:**

Microservices work independently, so you don’t have to change the codebase in order to modify the features. You can change one component, test, and then deploy it individually. In turn, you will deliver the app faster.

* **Evolutionary:**

Microservice architecture is a perfect choice for developers unable to predict the kinds of devices the app is going to run on. Developers can provide fast and controlled upgrades through not slowing down or stopping the apps.

While microservices offer the opportunity for better productivity and tool selection, cons include teams using different coding languages, frameworks, libraries.  This can paralyze the team if they’re not ready for such independence. But if you’re working on a large-scale complex app, microservice architecture is the perfect choice.

* **WHERE MICROSERVICES ARE IN USE THESE DAYS:**

Many organizations have already successfully embarked on their journey to the microservices world. In this section, we will examine some of the frontrunners on the microservices space to analyze why they did what they did and how they did it.

* **Netflix** ([www.netflix.com](http://www.netflix.com/)): Netflix, an international on-demand media streaming company, is a pioneer in the microservices space. Netflix transformed their large pool of developers developing traditional monolithic code to smaller development teams producing microservices. These microservices work together to stream digital media to millions of Netflix customers. At Netflix, engineers started with monolithic, went through the pain, and then broke the application into smaller units that are loosely coupled and aligned to the business capability.
* **Uber** ([www.uber.com](http://www.uber.com/)): Uber, an international transportation network company, began in 2008 with a monolithic architecture with a single code base. All services were embedded into the monolithic application. When Uber expanded their business from one city to multiple cities, the challenges started. Uber then moved to SOA-based architecture by breaking the system into smaller independent units. Each module was given to different teams and empowered them to choose their language, framework, and database. Uber has many microservices deployed in their ecosystem using RPC and REST.
* **Nike** ([www.nike.com](http://www.nike.com/)): Nike, the world leader in apparel and footwear, transformed their monolithic applications to microservices. Similarly to many other organizations, Nike too was run with age-old legacy applications that were hardly stable. In their journey, Nike moved to heavyweight commercial products with an objective to stabilize legacy applications but ended up in monolithic applications that were expensive to scale, had long release cycles, and needed too much manual work to deploy and manage applications. Later, Nike moved to a microservices-based architecture that brought down the development cycle considerably.
* **Twitter** ([www.twitter.com](http://www.twitter.com/)): Twitter, one of the largest social websites, began with a three-tiered monolithic rails application in the mid 2000s. Later, when Twitter experienced growth in its user base, they went through an architecture-refactoring cycle. With this refactoring, Twitter moved away from a typical web application to an API-based even driven core. Twitter uses Scala and Java to develop microservices with polyglot persistence.

* **CONCLUSION:**

Microservices are the most suitable for large-scale applications. Smaller apps are usually better off with a monolithic code base, though.

While it’s easier to develop and maintain independent microservices, network management requires additional efforts. Container platforms, DevOps practices, and cloud computing can help a lot in adopting the microservices architecture.