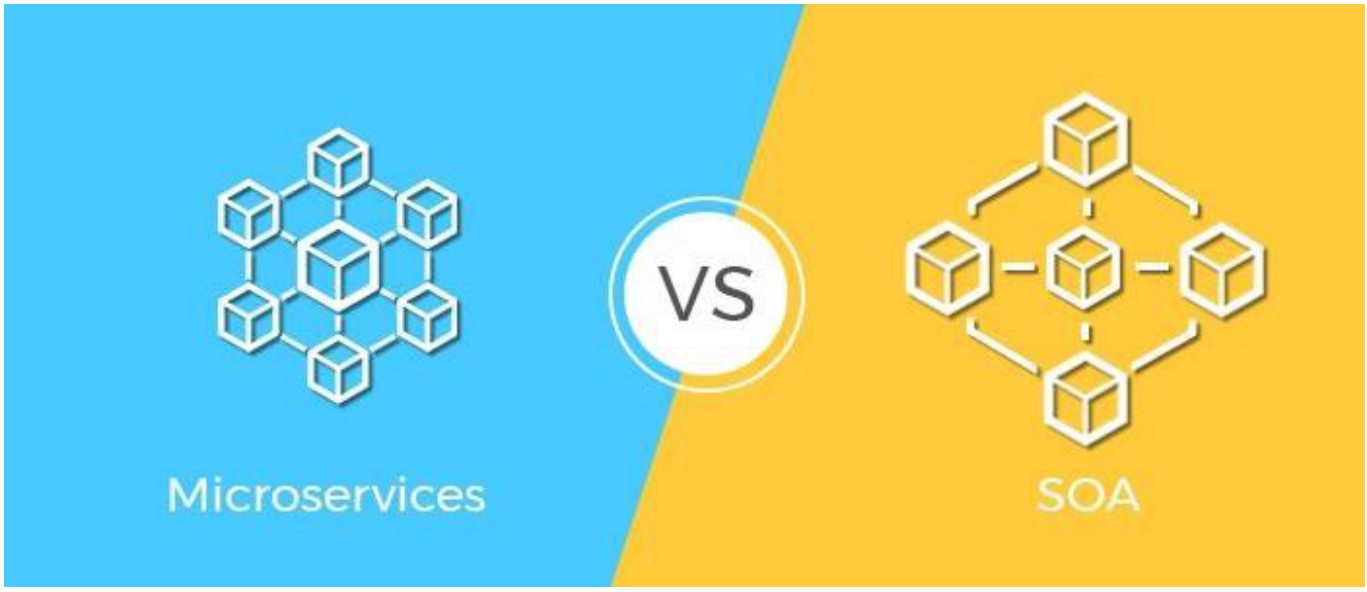
Daffodil Software

225 Followers · About Follow

Microservices vs Service-Oriented Architecture (SOA): Fundamental Differences

Daffodil Software Nov 29, 2018 · 5 min read



With increasing complexity and demand for highly scalable and robust applications, the conventional monolithic architecture is no longer the best choice. After a certain threshold, monolithic architecture tends to hinder the application performance and scalability. Moreover, with an enormous codebase, making changes to the tightly coupled, dependent processes in the monolithic architecture drastically increases the impact of single process failure.

which says: Gather together those things that change for the same reasons, and separate those things that change for different reasons. Eventually, Service-Oriented Architecture (SOA) and Microservices Architecture gained acceptance and enabled developers to build applications as a suite of small, de-coupled services, running in their own environments and are independently deployable.

While Microservices and SOA have application modularity in common, they differ in how the services are deployed and the size of modules. Before we understand the fundamental differences between the two- Microservices vs SOA, let's understand these architectures in detail.

Microservices: An Overview

Microservices is a software modularization approach that aims at dividing larger software system into smaller components. With a microservices architecture, an application is built with independent group of components that run each application process as a service. Such an architecture makes it easier for applications to scale, accelerate development process, offers space for experimentation, and reduce time-to-market for new features.

Microservices Architecture: Benefits

- Continuous Development and Deployment

Each component in a microservices architecture is developed, deployed, operated, and scaled independently, i.e. a change in one service does not affect the functioning of other services. Moreover, these services do not share code or implementation with other services. This ensures continuous development and deployment of large, complex applications.

- Eliminates Technology Commitment

Microservices can be created using different technologies (frameworks, languages, or OS). Therefore, it eliminates any long-term commitment and dependability on a

Since all services in an application are independent of each other, any halt in a process does not affect another. For example: Memory leak in a particular service will not have any impact on another service, ensuring that rest of the services keep receiving and handling incoming requests. If this is compared to a monolithic architecture, a single misbehaving component affects the entire system.

Large Scale Websites are using Microservices Architecture:

Realizing these benefits, organization are moving from monolithic architecture to microservices. For instance, Netflix, one of the most popular video streaming service is one of them. It handles over a billion calls per day on their video streaming API, from over 800 kind of devices. Each API call is then spread out to an average of six calls to backend services.

Microservices are being used to re-architect existing applications as much as for brand new project, according to Red Hat 2017 Microservices Survey. (Click to Tweet)

Service Oriented Architecture (SOA): Overview

Similar to Microservices, SOA is a software modularization approach, an architecture based on disparate services. Herein, a service is a well-defined, self-contained functionality. In the SOA architecture, different services communicate with each other to execute a functionality.

The SOA services are loosely-coupled, i.e. a service interacting with another might not know the technical details of another service. An SOA service consists of two fundamental components: Service Consumer that requests for some service and Service Provider that returns the result for the request.

Since Microservices and SOA enable developers to build independent services to build an application; they more or less share similar set of benefits. However, there are some basic differences between the two, which will be shared in the later segment.

Large Scale Websites are using SOA:

web app calls 100–150 services to get the data that is used to build a web page.

Microservices vs SOA: Differences

In both the architectures, each service has a certain responsibility, unlike a monolithic architecture. Thus, developers have the flexibility to develop the services in different technologies, which enables the team to choose the most relevant use-case fit for a service. However, both the architectures differ in following terms:

- Service Deployment and Scalability

The service deployment can be assigned and managed within multiple teams. However, each member in the team needs to know the common communication mechanism in SOA, which is not a dependency while building an application with microservices architecture.

In case of microservices, the services can be operated and deployed independently, unlike SOA. Therefore, is it easier to deploy new version of services frequently and thus scale the application accordingly.

Microservices vs SOA, another point of differentiation is fault toleration. In SOA, an Enterprise Service Bus (ESB) can be a single point of failure, which impacts the functioning of entire application. An Enterprise Service Bus (ESB) implements communication between mutually interacting applications in SOA. It can perform variety of functions, ranging from message transformation, routing, logging, protocol transformation, acting as security gateway etc. If one of the ESB service slows down, it will clog up the requests for that service. In comparison to SOA, microservices are fault tolerant as all services and their deployment is independent of each other.

In Service-Oriented Architecture, the services share data storage, while in microservices, each service can have an independent data storage. Data storage brings its own set of advantages and disadvantages. For example: While the data can be reused between all services, it can bring dependency between services due to tight-coupling.

SOA can either be monolithic or can have multiple microservices.

ALSO READ: Advantages of Developing Cloud Native Application on AWS

Microservices vs SOA: Which Application Architecture to Choose?

The above segment detailed the differences between Microservices and SOA. However, if you're still confused about which architecture will work best for your software development, get in touch with our tech-experts through our no-obligation 30 minute consultation program.

Originally published at insights.daffodilsw.com.

Microservices Soa Software Architecture Application Development Mobile App Development