Microservices Architecture?

While there is no standard, formal definition of microservices, there are certain characteristics that help us identify the style.  Essentially, microservice architecture is a method of developing software applications as a suite of independently deployable, small, modular services in which each service runs a unique process and communicates through a well-defined, lightweight mechanism to serve a business goal.  
   
How the services communicate with each other depends on your application’s requirements, but many developers use HTTP/REST with JSON or [Protobuf](https://github.com/google/protobuf/" \t "_blank).  DevOps professionals are, of course, free to choose any communication protocol they deem suitable, but in most situations, [REST](http://blog.smartbear.com/apis/understanding-soap-and-rest-basics/) (Representational State Transfer) is a useful integration method because of its comparatively lower complexity over other protocols.  
   
To begin to understand microservices architecture, it helps to consider its opposite: the monolithic architectural style.  Unlike microservices, a monolith application is always built as a single, autonomous unit.  In a client-server model, the server-side application is a monolith that handles the HTTP requests, executes logic, and retrieves/updates the data in the underlying database.  The problem with a monolithic architecture, though, is that all change cycles usually end up being tied to one another.  A modification made to a small section of an application might require building and deploying an entirely new version.  If you need to scale specific functions of an application, you may have to scale the entire application instead of just the desired components.  This is where [creating microservices](https://smartbear.com/learn/api-design/how-to-create-a-microservices-setup/) can come to the rescue.

## **SOA vs. Microservices**

“Wait a minute,” some of you may be murmuring over your morning coffee, “isn’t this just another name for SOA?”  Service-Oriented Architecture (SOA) sprung up during the first few years of this century, and microservice architecture ([abbreviated by some](http://blog.xebia.com/2015/02/04/microservices-versus-the-common-soa-implementation/) as MSA) bears a number of similarities.  Traditional SOA, however, is a broader framework and can mean [a wide variety of things](http://martinfowler.com/bliki/ServiceOrientedAmbiguity.html).  Some microservices advocates reject the SOA tag altogether, while others consider microservices to be simply an ideal, refined form of SOA.  In any event, we think there are clear enough differences to justify a distinct “microservice” concept (at least as a special form of SOA, as we’ll illustrate later).  
   
The typical SOA model, for example, usually has more dependent [ESBs](http://en.wikipedia.org/wiki/Enterprise_service_bus), with microservices using faster messaging mechanisms.  SOA also focuses on imperative programming, whereas microservices architecture focuses on a responsive-actor programming style.  Moreover, SOA models tend to have an outsized relational database, while microservices frequently use NoSQL or micro-SQL databases (which can be connected to conventional databases).  But the real difference has to do with the architecture methods used to arrive at an integrated set of services in the first place.   
   
Since everything changes in the digital world, [agile development techniques](https://smartbear.com/learn/software-testing/what-is-agile-testing/) that can keep up with the demands of software evolution are invaluable.  Most of the practices used in microservices architecture come from developers who have created software applications for large enterprise organizations, and who know that today’s end users expect dynamic yet consistent experiences across a wide range of devices.  Scalable, adaptable, modular, and quickly accessible cloud-based applications are in high demand.  And this has led many developers to change their approach.

## **Examples of Microservices**

As Martin Fowler [points out](http://martinfowler.com/articles/microservices.html), Netflix, eBay, Amazon, the UK Government Digital Service, realestate.com.au, Forward, Twitter, PayPal, Gilt, Bluemix, Soundcloud, The Guardian, and many other large-scale websites and applications have all evolved from monolithic to microservices architecture.  
   
Netflix has a widespread architecture that has evolved from monolithic to SOA.  It receives more than one billion calls every day, from more than 800 different types of devices, to its streaming-video API.  Each API call then prompts around five additional calls to the backend service.  
   
Amazon has also migrated to microservices.  They get countless calls from a variety of applications—including applications that manage the web service API as well as the website itself—which would have been simply impossible for their old, two-tiered architecture to handle.  
   
The auction site eBay is yet another example that has gone through the same transition.  Their core application comprises several autonomous applications, with each one executing the business logic for different function areas.