# **How is Microservices different from monolithic?**

Most of the big companies and start-ups initially build their systems using monolithic architecture to set up business faster and get moving. After some time due to mature in projects and rapid growth the code gets complicated which leads to complex architecture to maintain it. At the same time, business may lose speed, flexibility and agility which makes it harder to react to the market needs.

There comes the microservice approach to answer problems of large and complex IT systems to develop their applications as a suite of small services which communicate each other via API gateways.

Benefits of micro services are they can be developed, scaled and deployed independently by specialized teams. Changes to a particular microservice can be done and deployed separately without disturbing the other parts of the system.

Thereby many companies are transforming their IT infrastructures into a microservice architecture. Amazon and Netflix went through the process of transition and paved a way for their followers.

**What led Amazon to transform from monolith to microservices??**

Upto 2001 the amazon.com retail website was a large architectural monolith.

Earlier in 2000 at Amazon when there were lot of in-process changes from hundreds of developers, the conflicts are resolved and merged them into a single master version that waits on the queue to be moved into production. Even that single big large new version adds a lot of overhead on its delivery pipeline because that whole new code needs to be rebuilt, later all the test cases need to be re-run and then the entire application need to be deployed in production unit.

Amazon used microservices to simplify and shorten the pipeline.

Consider a single purpose served functional units. For example, there was a single service that rendered the “Buy” button on the retailer’s product detail pages. Another had the function of calculating the correct tax during checkout. At the time of their creation, there single-purpose functions seemed to be simple enough to achieve. But imagine hundreds of development teams single-purpose functions need to be merged together week after week, month after month as the fabric of SDLC became bigger and bulkier.

Amazon used decoupled architecture that made functions to communicate with the rest of the world through their own web service APIs.

The decoupling of services enabled automated deployments systems using pipeline model. But still it took a long time for a code change to go from a developer check-in to be running in production where customers could use it because of manual handoffs between departments to introduce process integrity. This lead to deadtime (intervals where nothing was happening), inefficiencies, wasted space and long queues.

This lead to include inline scripting of events and integration with private repositories in the deployment pipeline.