Micro-services based strategy promises to break open the "black box", unlock the rigidity and enable flexibility by allowing deployment of self-contained & independent services.

What are Microservices?

Micro-services are the opposite end of the spectrum as compared to monolithic systems. While Monoliths are massive software systems with complexity of highly interdependent, large complex code tucked away inside the proverbial “black box”. Micro-services based strategy promises to break open the "black box", unlock the rigidity and enable flexibility by allowing deployment of self-contained & independent services. Flexibility & speed is driven primarily by removing release dependencies between various functional subsystems. Microservices key attribute is that they are self-contained and independent, have a clearly defined interface and ability to have separate persistence data of itself (in some cases having their own database).

Benefits

Contrary to the popular belief prevalent several years ago that there is a utopian end state architecture, it is self-evident that there is no static end state architecture that will be able to deal with today’s continued fast paced Digital age and the demands of speed to market and business agility this new age requires. Microservices allow you to deal with this need for speed and agility by allowing continuous evolution of the architecture and enable creation of an architecture ecosystem that allows changing of small subsystems within the larger architecture/functional eco-system over time and thereby avoiding fork-lift upgrade of legacy monoliths and major migrations. This flexibility in the end allows a faster time to market as it enables the deployment architecture and associated release mechanism to become less convoluted as the inter-dependencies that need to be reflected between various releases and services are reduced.
Keys to successfully implement Microservices

1. Agile & DevOps across cross functional teams need to be matured inside organization. These DevOps teams should have full ownership of their Microservices from conception & design to production (suggested team composition: engaged production owner, scrum master, 3 to 5 feature developers, 1 to 2 test developers, 1 delivery engineer)

2. There will be an increased complexity in testing as there are many permutations & combination of services to deal with hence automation and intelligent testing strategies are very important (Automated Continuous Integration, Deployment and Testing)

3. Team must use advanced software techniques to create the microservice including: trunk based development, cloud native design, API versioning, backwards compatibility, and front ends must use feature toggling

4. Artificial Intelligence & Machine Learning: The long term benefits of attaching AI/ML to each microservice can be a game changer, hence a game-plan to attach AI/ML to your microservices should not be overlooked

How to get started:

- Start with a pilot not big-bang
- Pilot should start in a domain where there is organization maturity & sponsorship to implement Full CI/CD and DevOps methodology – test & learn continuously
- Pick a domain with business-win, where agility & time to market can be measured
- Conduct team kick off to ensure proper Agile methodology (requires upfront prep time to change work intake)
- Microservices will require investment and returns may not be apparent in short time, so go for an area that can take some level of investment and an organization that can appreciate incremental improvements in the Microservices based architecture.
- Adding a functionality to Monolith might be cheaper in the short term but in the long term microservices benefits will outweigh short term cost advantages of Monolith custom changes by delivering faster time to market & more business agility