newt|global
Aiming At Customer Delight

info@newtglobal.com

CONTINUOUS INTEGRATION & CONTINUOUS DELIVERY

MICROSERVICES IN AND OUT

Organization should be culturally aligned, as well as provide a subtle environment in adopting to a Micro Services architecture. Transitioning or Developing applications using Micro Services architecture is definitely not a cake walk. While the popularity of Micro Services is high, developers and testers really find it difficult in transitioning a monolithic style application to a micro services build architecture. This popularity is partly off the back of trends such as Cloud, DevOps and Continuous Delivery coming together as enablers for this kind of approach, and partly off the back as great companies have implemented successful micro services architecture that is well suited for their business.

newt|global
Aiming At Customer Delight

info@newtglobal.com

AN INTRODUCTION

Over the past decade, Mobile Technology has replaced many traditional devices, processes, lifestyle and way of doing business. While Mobile landscape has increased the scope of doing business and has created a new gateway to an era of apps, we must also understand how deeply it has penetrated into the life of each individual. Accidents have increased due to mobile text-drive, speak-drive, and radiations from the telecommunications tower have proven to harmful to living beings. Setting up mobile platform involves huge operations and effort. Having said this, the growth of Mobility does not decline after have seen the business benefits reaped out of it.

Micro Services are one of those ideas that are nice to be implemented, at the same time involves high end complexity when it meets reality. The pre-requisite to build applications' using Micro Services architecture that an organization should be prepared and that that the developer, tester, architect and the project manager should be aware of –

- Network is homogenous, reliable and secure with zero latency
- DevOps culture. All incidents should involve ops and dev
- Rapid Application Development. Deploy quickly to test and production
- Complete Automation – Installation, DB upgrades, Configuration
- Rapid Provisioning. Servers and Environments are disposable
- Hosts are immutable

FACTS TO CONSIDER

Substantial Infrastructure Operations

The proponents of the Architecture enable all the tools and resources required to set up a Micro Service platform. While a Monolithic application is straight forward to deploy in a small server cluster, Micro Services talks about tens of separate services to build, test, deploy, integrate and run to a single server in a polyglot environment. These services essentially need the application server running full time with a high storage space that is network resilience and reluctant to downtime. If each service has its own database, then handling the association of data between different databases and availability of those databases 24/7 increases operations effort. Only when there is complete Automation in place to test and deploy a build, enablement of Micro service can be achieved. Automation here again talks about implementing set of tools and governance practice that would be required to set up a DevOps environment.

Workforce Up-Skill

As there are pre-requisite to set up an environment to build Micro Services, there requires necessary requisite skill for a Developer to possess in order to work on micro services. We are talking about upskilling or cross skilling the existing team or hire a set of resources with good knowledge in DevOps function. A Developer with just programming and database knowledge alone cannot build a micro service, while that is easily achievable to build a monolithic application. When we are talking about resources with DevOps experience, we are also talking about substantial amount of time and effort required to train these resources or hire them from outside world. Human Resource becomes a high order of magnitude to think through while implementing a Micro service architecture.

Imbedded Interfaces

When we are breaking systems into multiple components, we have to introduce explicit interfaces between them to have an integrated system. There will also be several external interfaces like the data storage, payment functions, etc. Hence the message format on all the services have to semantically same to understand the

syntax and communicate with the each other. Since, there would be separate micro service for each of the function, at times we might have to ignore the fact of phase wise release as any of the function need not to have to be integrated during release A or B. In Micro services, Interfaces act as contracts and changing contract or having no contract requires changing multiple services.

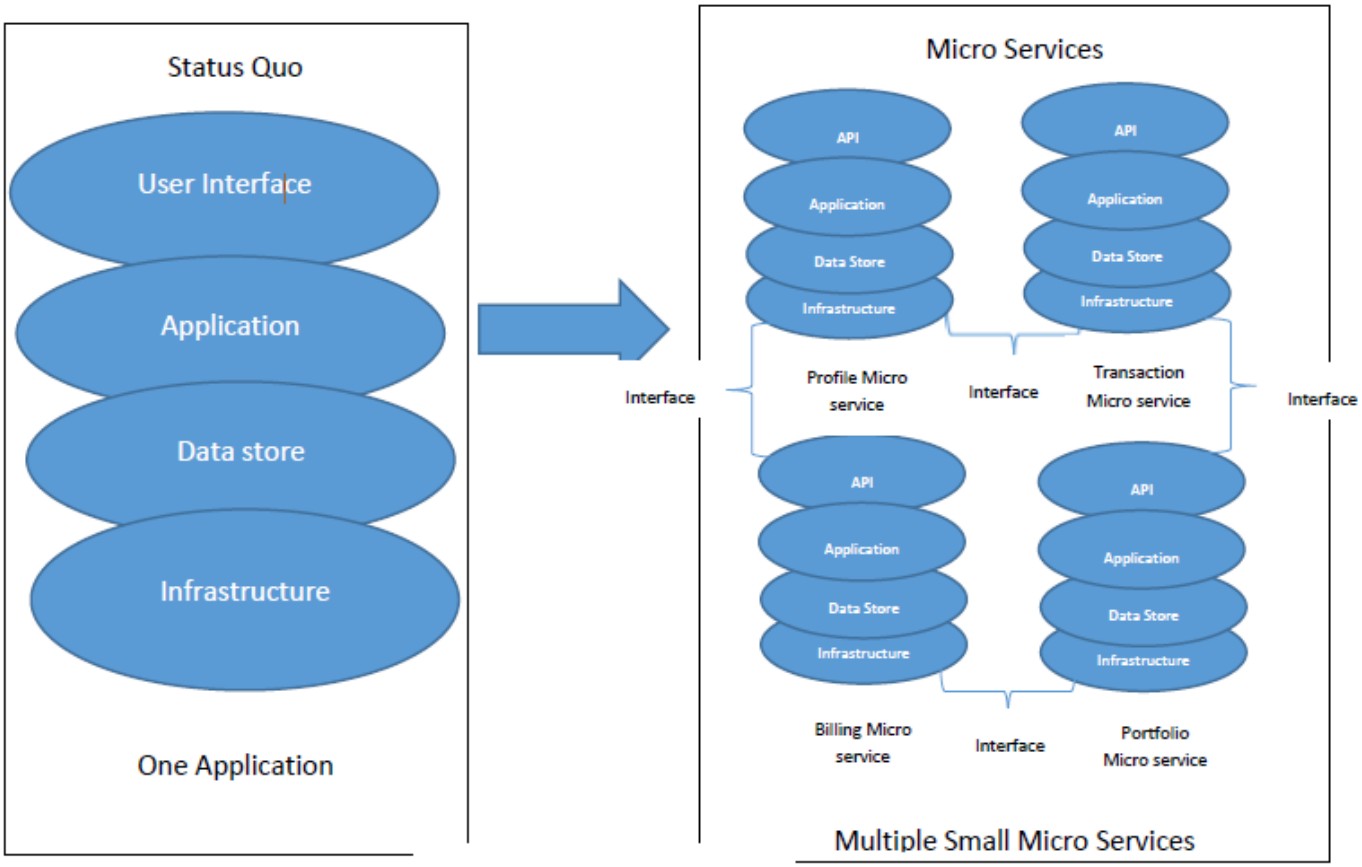

Fig.: Interfaces in Micro Services

Effort Duplication

Assume that there is a new requirement to change the way price is calculated for a certain product line. In a typical Micro Service environment, we introduce a new service and allow other services to call wherever needed to append this requirement. We could not take this call lightly, as there is a potential synchronous coupling into the system. In order to avoid the coupling effect, we would have to duplicate the effort by introducing the price change in all of the services. In addition to the duplicate effort, we also increase the effort of testing and

maintenance going forward. Or, there is also other option of sharing the resources between the services to make sure the new change is coming into effect. In a Monolithic style application, it will be just writing or appending a piece of code and making it available through the application irrespective of the services and calls.

The first step before building an application using Micro Service architecture is to have setup web services using Restful API. SOAP based web services are common in use in a Monolithic application. If the transition has to happen from a monolithic application that uses SOAP UI, then we would need to refactor the SOAP interfaces to entity based REST interfaces. Micro service imply a distributed system and hence we introduce a lot of remote procedure calls, REST API's or messaging to stitch the components across various processes and different servers.

Testing Complexity

It can be difficult to create a sandbox test environment for each of the service in manual or automated manner. With Asynchronicity and dynamic message load, the test systems fails to get confidence of releasing the build to production without testing the different scenarios. Micro Services architecture built with SPRING BOOT framework emphasis on monitoring the app before reaching production that however fails to detect bugs and functional changes to the environment. However, testing micro service at an isolated environment helps to achieve a bug free application at that unit or component level, but when it comes to integration testing with a dynamically changing environment, the test team have to pre-occupied with all the tools, processes and automation in place.

CONCLUSION

Currently, not any service provider or system integrator provide a complete framework and open source the tooling part to support a micro service from an operational perspective. It's likely therefore that a team or organization rolling out Micro services will need to make significant investment in custom scripting or development to manage these processes before they write a line of code that delivers business value and make sure there is Automated Testing place at each phase start from the Design till deployment.