Microservices Basics

# Why MICROSERVICES?

Companies like Netflix, Amazon has adapted the concept of Microservices in their products. Microservices is the hottest topic in software industry. Many organization wants to adopt it.

Devops can plays very well with Micro services.

But what is MicroService?

why organization should adopt this?

To understand it let take a look at the Monolithic Software.

In Monolithic Software mainly we use 3-Tier architecture

1. Presentation layer
2. Business Layer.
3. Data Access Layer

From Browser client post a request, Business ness tier execute the business logic and from database it’s collect Persistence data.

However, the problem with this type of system is

All codes (**presentation, business layer and data Access layer**) are maintaining in a same code base. Logically we divide the services like JMS Service, Data-Access Service but they are in same code base and deployed as a single unit.

Even though you create a multi-module project still One Module is dependent on another and it needs dependent modules in its class path.

What I am trying to say although you use distributed environment but it runs under single process context

So,

In a single process different services are communicating with each other. To achieve this artifacts and required library(jars) are required in each Application Container.

Like a JMS service want to use Data Access layer so JMS Container needs Data Access layer jars and its dependent jars.

In this concept there are lots of **pain points** and this architecture is very rigid.

I am trying to explain the problems

**Problem 1:**

As there is one codebase it grows gradually, Every Programmer irrespective UI Developer or Business layer developer they commit in same code base so it is very inefficient to manage. Suppose one developer work in JMS module but he has to pull whole codebase to his local and configure whole module in order to run it local server. Why so? He should only concentrate for JMS Module, but current scenario stops him to do so.

**Problem 2:**

As there is one code base and modules are dependent with each other, minimal change in one module need to generate all artifacts and need to deploy in each server pool in distributed environment.

Suppose in a multi-module project JMS Module, Business Module dependent on Data Access Module so a simple change in Data Access Module, we need to re-package all JMS Module and Business Module and deploy them in their server pool.

**Problem 3**:

As in Monolithic software uses 3 -Tier Architecture, so 3 cross functional teams are involved to develop a feature so although 3-Tier Architecture made for separation of responsibility but in a long go responsibility boundary are crossed unintentionally and layer are losing it fluidity and become rigid.

Suppose inventory management feature has to develop so, UI, Business layer and Data Access layer

each has their own job but everyone wants to take control of the main business part so that when defects come they can solve it and not dependent on another layer developer. Due to this competition Some time boundary has been crossed result an inefficient architecture.

**Problem 4 :**

In many projects I have seen there is Developer team and another is support team developer team only develops and after release the feature to production they handover it to Support team. I personally don’t support this culture although some Knowledge Transfer happens during handover but that not solves the problem. For a critical incident Support team seek helps from developer team which is disastrous and support team reputation gone down.

**Problem 5:**

As our system is Monolithic same our team management. Often We create team base on the tier

Like UI developers, Back end developers, DB programmer and they are expertise in their domain but have least knowledge about other layers so for a critical problem which expands over all layer blame games starts which is annoying not only that it takes several time to decide which layers’ problem it is and who will come forward and solve the issue.

Netflix, Amazon address this problem comes a solution name Microservices.

Microservice tells us to brake product or project in to independent services so that it can be deployed

and mange solely and not depending on any other services.

After seeing this definition obvious question arise in which basis I break down my project in to independent services?

Many people thought wrong about microservices . Microservice not telling about brake your project based on Tier like one JMS service , UI Servce, Logging Service .etc

No this is absolutely not, we need to brake it function wise . A complete functionality and this functionality may consist of UI, Business, Logging, JMS, data Access, Jndi look up service etc.

***Function should not be divisible and not dependent on other function.***

So If project has Inventory, Order, Billing, Shipping, UI shopping cart modules, so we can brake each services as an independent deployable module each has own maintenance, monitoring, application servers.

**The most important culture of Micro-service is Whoever develop the Service it is that teams responsibility to manage that service so no handover concept.**

**This Microservice has many more benefits**

**Benefit 1:**

As in monolithic software you only develop in one language say java as codebase is single but in Microservice as each service is independent and each service is a new project .Each service can be developed in any language which is optimum for this requirement.

**Benefit 2:**

Developer are only concentrate on particular service so codebase will be very small and developer has intense knowledge over that service

**Benefit 3**

May be one service needed to talk with another service so they can talk via API or specifically by Rest service. As Rest service is the medium to communicate so there is little transformation happens so I can say unlike SOA , Micro-service Message bus is much more thin than ESB . where ESB doing lots of transformation, categorization micro- service message bus does less thing.

Lot of people think SOA and Microservices are same thing although by definition they are look like same but SOA used for communicate different system over a ESB(Enterprise Service Bus) where ESB taking lot of responsibility to manage data, transformation of input one form to another, do categorization, use BPEL so I can say ESB is very intelligent and big module

But Microservice only use a dumb Message Bus which just transfer the input from one service to another

As Microservice communicate through REST so transformation scope is very less only one service is dependent on another service via API call.

**Microservice itself has some shortcomings**

As every functional aspect is an individual service so in a big project there are many services so monitoring this services add a new overhead.

Not only that for a service failure ,to track down which service is root cause for that is also a pain staking job.

Each service generates log so there is no central log monitoring log also a painful stuff we need a very good log management system for that.

In microservice each services communicate through API calls/remote call which has more overhead than monolithic software’s inter process communication call.

In spite of that Microservice do the real separation of responsibility so I believe to do efficient work, Organization has to adopt Microservices.