International Research Journal of Computer Science (IRJCS) ISSN: 2393-9842
Issue 05, Volume 4 (May 2017) SPECIAL ISSUE www.irjcs.com

Microservices Architecture

Priyadarshini S[#1*1], Shilpa G.V[*2*2]
UG Student & AssistantProfessor
Department of Computer Science & Engineering,
Vemana Institute of Technology, Bangalore, Karnataka

Abstract: ***Nowadays, software development team prefers to use Microservices architecture for building complex web application. Microservices is an architectural style which are used to build complex applications formed by small independent, decoupled process. Each of micro services mainly focus on completing one task and contains own database. Microservices are composed together to form a complex application and each Microservices can be developed using different programing language. Microservices can be deployed independently and run on its own computing unit. In Microservices architecture communicates with individual's service mainly over HTTPS/REST protocol.***

Keywords: ***Microservices***

I. INTRODUCTION

Languages such as Java, C/C++, Python which are used to development of server-side applications, provides abstraction to break down the complexity of program into module. However their modularisations depend on sharing of resources of same machine therefore the lack independent execution and such architecture is called as monolithic architecture. As impact we can see that when new developers join a large project team, they require certain amount of time to become used to the code base. Developers are hesitant to make enhance application in fear of breaking something else because of unfamiliar dependency. Microservice application make developer work easier by braking their work into small independence teams and to integrate that work as it is delivered.

II. WHAT IS MONOLITH?

Monolith is software application whose modules cannot be executed independently. This makes difficult to maintain and reuse.

III. WHAT ARE CHALLENGES WITH MONOLITHIC ARCHITECTURE?

There are certain challenges of monolithic architecture are Code complexity and maintainability, Deployment becomes the bottleneck, Fear to change Lack of ownership, Failure dependencies One size doesn't fit all (ex: relational DB) Hard to scale out.

IV: WHAT IS MICROSERVICES?

Microservices is an implementation approach for service-oriented architectures which is used to build independent, flexible, decentralized in order make developer work easier by braking their work into small independent teams and to integrate their smaller service into final application. Developers are independent to use programming language of their choice for development and finally expose their application as web service. In Microservice architecture, one application will be based on several Microservice each performing one independent task of application. As service are distributed and loosely coupled scaling and design of individuals service can be made without affecting overall application. Example for Microservice: Consider a cab booking application example Ola or Uber mobile based application. Cab booking application consists of several feature/functionalities such as Maps, online payment, customer feedback, customer information maintaince etc. Instead of developing each feature natively in the one application, already available service can be consumed as web service. For Maps, Google maps can be used which is already available as web service. For Online payment, PAYTM service can be used and integrated with application. Similarly already existing service can be used to build new application which will reduce the development and maintenance time and effort. With Microservice based architecture, any service can be easily unplugged and new service can be used with minimum effort example PAYTM service be removed and **free charge** service can be used for online payment

V: WHY WE HAVE TO GO FOR MICROSERVICES?

There are many benefits of Microservices that overcome the disadvantage of monolithic architecture:-

1. FASTER DEVELOPMENT AND DEPLOYMENT
2. AUTONOMY OF TEAMS, CULTURE OF CHANGE
3. OWNERSHIP AND DEVOPS CULTURE
4. COMPOSABILITY AND REUSABILITY
5. MORE MAINTAINABLE CODE
6. BETTER SCALING AND OPTIMIZATIONS
7. FAILURE ISOLATION AND RESILIENCY

VI: COMPARING MICROSERVICES AND MONOLITHIC ARCHITECTURES

MONOLATHIC ARCHITECTURE	MICROSERVICES ARCHITECTURE
A single code base for the entire application	Multiple code bases. Each microservice has its own code base.
Difficult to understand and often confusing and hard to maintain	Much better readability and easier to maintain.
Typically entirely developed in one programming language	Each microservice can be developed in a different programming language
Requires you to scale the entire application even though bottlenecks are localized	Enables you to scale bottle-necked services without scaling the entire application.
Complex deployments with maintenance windows and scheduled downtimes.	Enables us to measure bottle-necked services without mounting the entire application

VII: DEVELOPMENT LIFE CYCLE WITH TEAMS

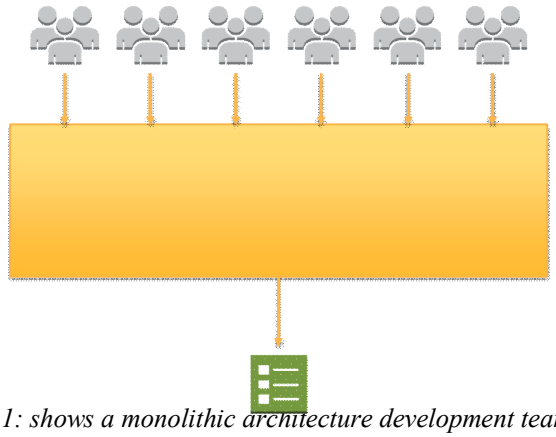

Fig1.1: shows a monolithic architecture development teams lifecycle

In monolithic architecture all team member focuses on the overall of the application. As impact we can see that when new developers join a large project team, they require certain amount of time to become used to the code base. Developers are hesitant to make enhance application in fear of breaking something else because of unfamiliar dependency. It consumes lot of time to deliver the project and when problems do occur hard to detect specifically where the problem occurred.

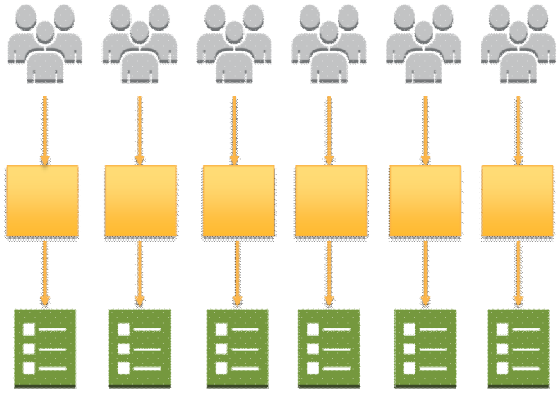

Fig1.2:shows a microservice architecture development teams lifecycle

Microservices allow the developer to divide the work units among the team and integration once when all unit of work done. Here each team focuses on single application and we can easily identify where the problem occur which make debugging more easy.

VIII: MICROSERVICES ARCHITECTURE CHARCTERISTICS

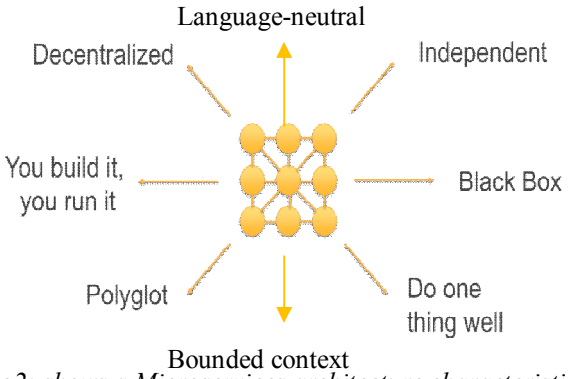

Fig2: shows a Microservices architecture characteristics

In a Microservices manage their own database so we can have multiple databases. Microservices is loosely coupled so level they share data should be as small as possible. We can create and destroy as new application emerges since they are loosely coupled. Do one thing well convey that we focus on the product than project.

IX: EXAMPLE OF MICROSERVICES

Many company uses microservices like Amazon, Netflix, Uber, eBay, Soundcloud, Twitter etc... Consider a cab booking application example Ola or Uber mobile based application. Cab booking application consists of several feature/functionalities such as Maps, online payment, customer feedback, customer information maintaince etc. Instead of developing each feature natively in the one application. Amazon now has moved to microservices architecture and they get many calls from variety of application that includes API that manages the web service, video API for streaming. Consider auction site eBay also undergo similar transition their core API consist of many autonomous application each one uses different execution logic for different functions.

X: SYSTEM ARCHITECTURE

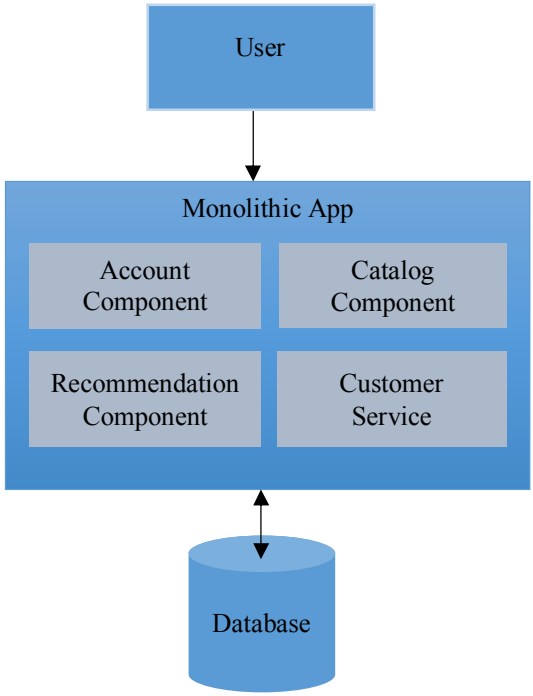

Fig3.1: shows a monolithic architecture with single database

In monolithic architecture they have single database to interact with. Debugging is difficult as we do not comes to know where actually fault lay. It is built using same programming languages and it if not flexible compared to microservices architecture.

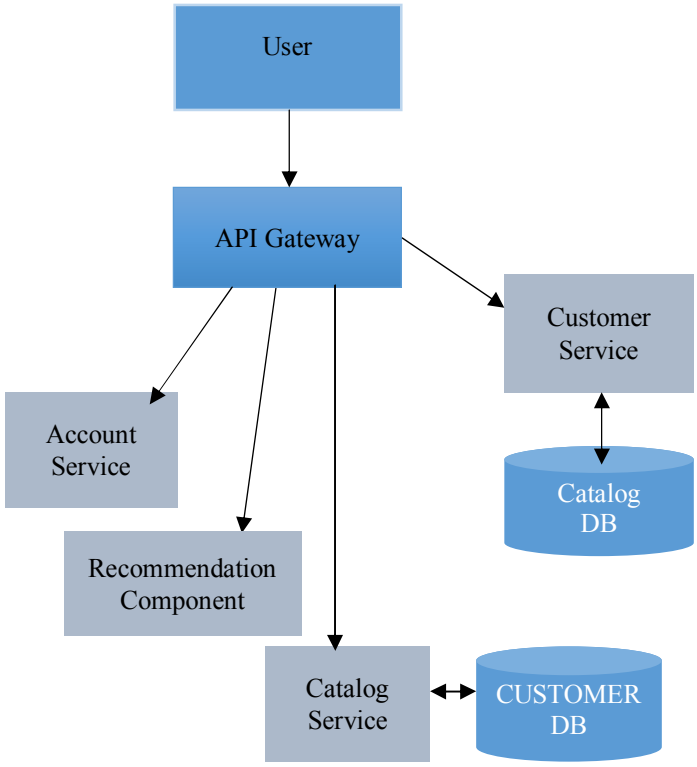

Fig3.2: shows a microservice architecture with multiple database

By adopting a microservices architectural style with a single platform development allow service management teams to support more easily multiple product and service teams. When problems occur in production, it is easier to identify and isolate the problem operations can also identify who from development team should be brought in to troubleshoot the problem.

XI: HOW DOES IT WORK?

Microservice architecture can be developed using any programming language. They will interact with other Microservices by using language neutral application program interface API like REST. Microservices don't need to know anything about underlying implementation of other microservices. When we deploy our application onto the cloud we will be given an end point URL using that we can access our application from anywhere, in this way we make one API to call another API. We can plug and play services just by changing end point URL.

XII: CONCLUSION

Microservice is an architectural style that can be formed by combination of numbers of application that are built using different type of languages. Instead of building application newly we use an application already built and use an API that calls another API in this way it allowsus to have a better, scalable and flexible application. It focuses on product than project. It uses decentralised govern and applications are highly decoupled. If any better application emerges than current application we can replace that application with better one and if in case anything goes wrong debugging is easy. IN this way Microservices provide the better approach to run the business.

REFERENCES:

[1]. Nicola Dragoni1, Saverio Giallorenzo2, Alberto Lluch Lafuente1 Manuel Mazzara3, Fabrizio Montesi4, Ruslan Mustafin3, Larisa Safina3,4, Microservices: yesterday, today, and tomorrow,2017
[2]. Daniel Escobar, Diana C´ardenas, Rolando Amarillo, Eddie Castro, Kelly Garc´es, Carlos Parra, Rubby Casallas. Towards the Understanding and Evolution of Monolithic Applications as Microservices.in 2015
[3]. Tasneem Salah, M. Jamal Zemerly, Chan Yeob Yeun, Mahmoud AI-Qutayri, Yousof AI-Hammadi, The Evolution of Distributed Systems Towards Microservices Architecture,2016
[4]. Edward B. Allen, Taghi M. Khoshgoftaar, and Yan Chen. Measuring coupling and cohesion of software modules: an information-theory approach. In Proceedings Seventh International Software Metrics Symposium, pages 124–134, 2001.

[5]. Alexey Bandura, Nikita Kurilenko, Manuel Mazzara, Victor Rivera, Larisa Safina, and Alexander Tchitchigin. Jolie community on the rise. In 9th IEEE International Conference on Service-Oriented Computing and Applications, SOCA, 2016.
[6]. Len Bass. Software architecture in practice. Pearson Education India, 2007.
[7]. Len Bass, Paulo Merson, and Liam O'Brien. Quality attributes and service-oriented architectures. Department of Defense, Technical Report September, 2005.
[8]. James M. Bieman and Byung-Kyoo Kang. Cohesion and reuse in an object-oriented system. In Proceedings of the 1995 Symposium on Software Reusability, SSR '95, pages 259–262, New York, NY, USA, 1995. ACM.
[9]. Andrew Birrell, Greg Nelson, Susan Owicki, and Edward Wobber. Network objects. SIGOPS Oper. Syst. Rev., 27(5):217–230, 1993.

