Microservices As an Application Architecture

Damian Nolan

***Abstract*—Microservice Architecture is an approach in software engineering and design that aims to divide an application into small constituents that are highly cohesive and loosely coupled. The following paper is intended to give an overview and exploration into the architectural style. Microservice Architecture has gained extensive exposure in the last number of years - Throughout this paper we will introduce Microservice Architecture as a concept, discuss Microservices as a subset of Service Orientated Architecture (SOA), compare and contrast Microservices with traditional Monolithic Architecture and review Docker, as a containerization platform for building Microservice applications.**

I. Introduction

THE term "Microservice" does not hold a true or set definition. It was merely a term introduced by a number of software architects at a workshop near Venice in May, 2011 to give a context to a number of reoccurring design principles and patterns that seemed to be emerging more frequently. Just under a year later, James Lewis presented at 33rd Degree in Krakow where he spoke about building systems composed of systems and focusing on the Unix philosophy of small and simple. [**JamesLewis33rdDegree**] And so, with the introduction of a hot new buzzword on the scene, "Microservices" began to get more and more attention.

Martin Fowler describes the Microservice Architectural style as a approach to developing a single application as a suite of independently deployable services each running in its own process. [**MicroservicesResourceGuide**] Each service should be independent in its our right and provide functionality based around a single responsibility. Services should be loosely coupled, provide high cohesion and adhere to the single responsibility principle. All of a sudden we can see that many of the traits we mentioned are key concepts in Object Orientated Programming. This is common in design principles of Service Orientated Architecture. Stubbings and Polovina [**StubbingsPolovina**] explore and contrast the how Object Orientated expertise can be leveraged in the design of Service Orientated Architecture (SOA).

II. Are Microservices just SOA?

This immediately sparks the debate - Is Microservice Architecture really just SOA? Upon the rise of the term "Microservices" into the community, many SOA architects and engineers were coming forward, saying "We've been doing this for years!". And as Martin Fowler mentions in his keynote[**GOTOConference**], Service Orientated Architecture is a very broad term. It encompasses a vast landscape of design concepts, principles, patterns and implementation standards. Fowler goes on to mention that in essence Microservices can be really be considered a subset of SOA.

Indeed, this is a popular and well defined statement as it is backed up by Adrian Cockcroft, the man responsible for pioneering the architectural style at Netflix as they moved towards cloud based and Microservice orientated design. He describes the architecture as "fine-grained SOA", in his keynote at Dockercon in 2014. [**adriancockcroft**]

Netflix are widely considered to be part of the pioneers of the Microservice Architectural style and have helped pave the way for it to flourish, introducing such libraries and tools as Hystrix[**Hystrix**] and Chaos Monkey[**ChaosMonkey**] for testing and strengthening systems.

III. Monolith vs MicroServices

Josh Evans of Netflix provides the analogy of the human body to describe the architectural style in his keynote. [**MasteringChaosNetflix**] He describes Microservices as the organs which make up the human body and work together as one organism. Perhaps the best way to put Microservices into context is by comparing the differences in contrast to a traditional monolithic architecture. Martin Fowler uses a similar example in his Resource Guide to Microservices where he outlines a potential structure of a traditional enterprise application. [**MicroservicesResourceGuide**] Take for example a simple application that exists in 3 tiers.

A. Presentation Layer

A Presentation Layer sits at the top of the application. This is the entry point of an application from a customer or user perspective, more commonly referred to as the front end. For example, this layer may consist of a web application composed of HTML, CSS and Javascript presenting a user interface to the client.

B. Logic Layer

A Logic Layer lies behind the user interface and provides functionality to carry out business operations. This could be a wide variety of functionality and responsibilities could be anything from user accounts to instant messaging to calendar events. For this purpose, lets take a Java server application as the example. Requests may be issues over HTTP and handled accordingly by the server application.

C. Data Layer

The Data Layer lives at base of the application stack. All of the raw data needed and in use by the other layers of

the application exists here. A data store such as a relational database could be employed to persist and manage data in this layer.

The monolith in this example is the server application. A monolith is a software application whose modules cannot be executed independently [**MicroservicesYesterdayTodayTomorrow**]. In the case of the example defined above we may need to make improvements, changes or simple bug fixes to any one of the services we described. Changes made to the source code of the application imply - rebuild and redeploy. But what if the changes are small and confined to one service? There is an overhead associated with this as it becomes expensive to build a new release for production and this may only be affordable every couple of weeks. Continuous delivery and continuous deployment are considered hidden dividends of Microservices [**hiddendividends**]. This allows developers to focus on developing and can speed up productivity across teams.

IV. Microservices Characteristics

Characteristics go here. - cite Building Microservices - Sam Newmann So how does one define a single Microservice? Or the architectural style as a whole?

V. Conclusion

The conclusion goes here.