Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Diploma Thesis

Microservices and Serverless architecture in web application

Apu Md Foyjur Rahman

Microservices and Serverless architecture in web application

Abstract

The terminology "Microservice Architecture" has jumped over the decade a particular way of designing/architecting software applications as a bundle of independently deployable service. Microservice is not a new term, but it is an evolutionary term/architecture to find out a pathway to do revolutionary things that already took place in cloud, web, server virtualization, mobile, containerization. This is an explicit responsive architecture to this amazing technology landscape that currently we live in. Alongside the term "Serverless Architecture" comes with a side by side to maximize and hassle-free automation in all life cycle of the application product. The Serverless architecture or computing offers the potential of application software in the cloud in the manner of auto-scaling, pay-as-you-go. This is a new trend in cloud computing, which enabled the business to get rid of underlying infrastructure issues.

My thesis aims to provide a more conclusive answer of how the Microservices and Serverless architecture reshape the current web application world.

In this study, we find out the difference between the classic monolithic and Microservices approach, Serverless architecture and standard backend approach, Single page and contemporary front-end approach, containers, and server hosting as well as FaaS and SaaS overview and some other third party services. We also conducted a single page application based on proposed architecture.

Keywords: Microservice, Serverless Architecture, Cloud, Docker, Virtualization, scalability, Single page application, Web Application, SPA.

Objectives and Methodology

Objectives

- Explain the difference between the monolithic approach and Microservices in web applications.
- Which reference architecture can best serve as the base for scalable web services?
- The principal research question, we derived the following specific questions:
    1. What aspects are influencing the adoption of Microservice Architecture?
    2. To what extend can containerization enhance the design and implementation of Microservice Architecture?
    3. To what extend can Microservice architecture improve the scalability of web services?
    4. To what extend can Microservice testing be automated?
- Develop and test a scalable Microservice Architecture for web services
- Describe the benefits of Serverless architecture in web applications.
- Describe single-page applications in Front-end development.
- Describe the benefits and uses of the Containers.
- Describe the various virtualization technologies.
- Prepare an overview and the model of the proposed application in Microservices and Serverless architecture.
- Implement the proposed application

Methodology

In the study, we employ two types of methods that are conducted concurrently. The research methodology will, however, inform the system development methodology.

Research Methodology

In these studies, a research methodology called Design Science Research Methodology (DSRM) is employed. The object that we propose is a scalable Microservice Architecture for Web Service. Research phases that have to be carried out in DSRM are (**Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007**):

1) Problem denotation & study(evaluation of current practice)

2) Determine the objectives of a solution (what would a better artifact accomplish?)

3) Prototype design & development

4) Prototype demonstration (finding a suitable context then use the artifact to solve problems)

5) Prototype evaluation (observing how effective it is in solving the problem)

6) Communication.

Problem Definition and Analysis

After this point, usually, the process iterates back to step (2) or (3). Following this DSRM method, we first investigate the market to gain insight into state of the art relating to Microservice Architecture (step 1 in DSRM). Based on the findings of this market analysis, we will identify issues associated with the platforms concerning scalability, which motivates the need for a new platform. Current technology used, architecture components, and functionality gaps will be acknowledged as well. Step (1) will be covered in chapter 1 and 2 in this report.

Defining objectives of a solution

In the next step (2), we will propose requirements and architecture components that need to be incorporated into the platform design based on the literature study. This phase is necessary to illustrate the inadequacy of solutions in the market in achieving our project goals. We will carry out a literature study on the topics of web application architecture, system-level virtualization, and scalability. In the practical portion, describe our implementation and steps needed to achieve the final working application.

Summary

The terminology "Microservice Architecture" has jumped over the decade a certain way of designing/architecting software applications as a bundle of independently deployable service. Microservice is not a new term but it is an evolutionary term/architecture to find out a pathway to do revolutionary things that already took place in cloud, web, server virtualization, mobile, containerization. This is an explicit responsive architecture to this amazing technology landscape that currently we live in. Alongside the term "Serverless Architecture" comes with a side by side to maximize and hassle-free automation in all life cycle of the application product. The Serverless architecture or computing offers the

potential of application software in the cloud in the manner of autoscaling, pay-as-you-go. This is a new trend in cloud computing which enabled the business to get rid of underlying infrastructure issues.

My thesis aims to provide a more conclusive answer of how the Microservices and Serverless architecture reshape the current web application world.

In this study, we find out the difference between the classic monolithic and Microservices approach, Serverless architecture and standard backend approach, Single page and contemporary front-end approach, containers, and server hosting as well as FAAS overview and some other third party services. We also conducted a single page application based on proposed architecture.

Conclusion

The introduction of Microservice Architecture has reinforced the resolve for agility in the software industry. Containerization is promising to transform IT in a more profound way than full virtualization. The scalability and the associated cost reduction and energy saving that can be achieved when the two technologies are applied concurrently is enormous.

The most crucial resource in IT is Humanware. However, it was discovered that for large project teams working on monolithic software, the man-month law works in reverse as you increase the number of developers. This means that the performance of the team is not proportional to the increased man-hours. Microservice Architecture enables scalable development of software since small manageable cross-functional teams can handle each Microservice.

Similarly, a monolithic application will scale horizontally by consuming more virtual machines or servers. Given that the monolith comprises software components whose functionality has varying demand from the users, it becomes wasteful to assign more resources to all software components equally. The Microservice Architecture addresses this problem by enabling scalability on a functional dimension. By splitting the application

into smaller units, the workforce per unit can be resized accordingly to enhance productivity. The number of technologies supported may be scaled accordingly as the need arises.

By employing distributed data stores in Microservice Architecture, scalability is enhanced by eliminating coherency delay that is prevalent in relational databases. The developer has the flexibility to choose the right database technology. Factors influencing the adoption of Microservice Architecture include virtualization, containerization, and the Internet of things.

To What Extent Can Containerization Enhance Design and Implementation of Microservice Architecture?

Containerization abstracts the complexity that is introduced by Microservice Architecture. Most functions that arise due to splitting a monolith into microservices such load balancing, health checks, etc. are handled at the orchestration layer. Similarly, the features such as service discovery, scheduling, inter-container communication are hidden from the developer and managed at the orchestration layer. The Docker Architecture is extensible through the use of plugins. Volume plugins allow third-party container data management solutions to provide data volumes for containers that operate on data, such as databases, queues, and key-value stores and other stateful applications that use the file system. Network plugins allow third-party container networking solutions to connect containers to container networks, making it easier for containers to talk to each other even if they are running on different machines. Docker Swarm is a powerful cluster management tool that is capable of handling over 1000 hosts and scheduling up to 30000 containers.

To What Extent Can Microservice Architecture Improve the Scalability of Web Services?

Scalability of web applications can be achieved through vertical scaling, horizontal scaling and functional scaling. By using Microservice Architecture it becomes possible to split a web application into smaller units that can be packaged as containers for efficient utilization of the hardware and software resources. With containerized microservices you can achieve a high software density in a data centre than using a virtual machine. This

means that scaling up and down a given function in a web application is easier, faster and less costly.

Scalability is multi-dimensional. Containerized Microservices makes functional scalability possible. Using Docker Compose, we managed to demonstrate how you can scale up service by specifying the number of containers using Command Line Interface. By varying the number of containers for a given microservice, we were able to achieve the desired scalability.

Microservice Architecture reduces friction amongst developer teams, operations team and quality assurance teams. As you scale up by adding developers to monolithic system the man-months increases. The coordination effort for a team of n members is proportional to n(n-1). By splitting the monolith into microservices you are able to assign 3-5 people to one microservice and this reduces friction though reduced coordination effort.

References

Adrian Cockcroft, (2014), ‑Migrating to Microservice‖. In: QCon London.

Agile Manifesto (Accessed on 3/07/2015) URL: http://www.agilemanifesto.org/

Alex Williams,(2016), ‑The Docker and Container Ecosystem eBook Series‖, The New Stack.

Alex Williams, (2014), ‑Flocker, A Nascent Docker Service For Making Containers Portable, Data and All.

Bob Williamson et al,(2015) ‑Akka in Action Manning‖.

Brendan et al.(2015), ― Borg, Omega, and Kubernetes, Lessons learned from three container-management systems over a decade‖, Google Inc.

Cloudsoft , ―Container Networking plugin‖, (accessed 17/07/2015) URL: http://www.projectcalico.org/learn

Cody Bumgardner, (2015), ― Openstack in action, Manning publications‖

Conway et al , (1968), ‒How do Committees Invent?‖, Datamation 14 (5): 28–31.

D. Ongaro , J. Ousterhout, (2014), ― In Search of an Understandable Consensus Algorithm In 2014 USENIX Annual Technical Conference, pages 305-319, Philadelphia,PA.

Damon Edwards, ‒What is DevOps?‖ (Accessed on 3/7/2015) http://dev2ops.org/blog/2010/2/22/what-is-DevOps.html

Danilo Poccia, (2016), ‒AWS Lambda in Action, Event-Driven Serverless Applications‖ First Edition, Manning .

David Hilley et al.(2009), ‒Cloud computing: A taxonomy of platform and infrastructure-level offerings‖.

Edward A. Lee, (2006), ‒The Problem with Threads‖ Technical Report No. UCB/EECS-2006-1

L. Bass, P. Clements, and R. Kazman, (1998), ‒*Software Architecture in Practice" ,* Addison Wesley, Reading, Mass.

L. G. Williams and C. U. Smith, (2004), ‒Web Application Scalability: A Model-Based Approach,‖ Proceedings of the Computer Measurement Group, Las Vegas.

L. Qian, Z. Luo, Y. Du, and L. Guo, (2009), ― Cloud computing: An overview. In Proceedi of the 1st International Conference on Cloud Computing, CloudCom '09, pages 626–6 Berlin, Heidelberg, Springer-Verlag.

Leonard Richardson, Sam Ruby, (2007), ‒RESTful Web Services‖, O'Relly.

M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A Patters A. Rabkin, I. Stoica, and M. Zaharia. (2009), ‒Above the clouds: A Berkeley view cloud computing‖, Technical Report UCB/EECS-2009-28, EECS Departme Universityof California, Berkeley.

Mahmood, Z., Hill, R. (2011). Cloud Computing for Enterprise Architectures. Computer Communications and Networks, Springer

Martin C. Roberts et (2016), ‒Agile Principles, Patterns, and Practices in C#‖, First Edition Prentice Hall .

Martin Fouler http://martinfowler.com/articles/Microservice.html accessed on 19-6-2015

Martin Fowler and James Lewis. Microservice (Accessed on 25/06/2015). 2014. URL:http://martinfowler.com/articles/Microservice.html

Mathijs Jeroen Scheepers, (2014), –Virtualization and Containerization of Application Infrastructure: A comparison‖ University of Twende.

Neil J. Gunther, (2007), –A Review of "Guerilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services, Performance Dynamics Company.

P. Mell and T. Grance. (2011), –The NIST definition of cloud computing‖, Technical report, National Institute of Standard and Technology - NIST.

Pasa Maharjan (2016), –Comparing and Measuring Network Event Dispatch Mechanisms in Virtual Hosts‖ Mater of Science Thesis, Tampere University of Technology.