International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 6, June 2020 ||

Tools and Techniques to build Scalable Systems

Deepak Kumar Singh[1], Dr. Manjunath M.[2]

PG Student, Department of Master of Computer Applications, R V College of Engineering®, Bengaluru, India [1]

Assistant Professor, Department of Master of Computer Application, R V College of Engineering®, Bengaluru, India [2]

ABSTRACT: As more and more people are getting connected to the internet, the development of applications have become more and more challenging as before the network technology like 3G[1] came along network traffic where not so high and only a few sites have to deal with such high volume of traffic. But now internet traffic is greater than before and even a simple website has to deal with lots of traffic. The previous model of application development ie. The monolithic applications are not highly scalable, are harder to maintain, and are not very cost-efficient. Now, the industry is moving to micro-service architecture which enables the new capability to develop highly scalable and maintainable systems and is more compatible with modern cloud infrastructure.

KEYWORDS: Microservice architecture, Cloud, Cloud services, Distributed design pattern

I. INTRODUCTION

Microservices architecture is based on the theory of Robert C. Martin who's approach of development was "gather together those things which change for the same reason, and separate those things that change for different reasons", this approach follows this fundamental principle and helps to develop highly scalable[2], decoupled applications. Microservice architecture has its roots in service-oriented architecture [3] where services are provided to components over the network and a number of services talk to one another either via passing data or two service coordinates one activity. Microservice helps to develop a light-weight solution by being as modular as the developer want to be, it is like small services to deployed separately and running its own process the advantage of having this level of decoupling is that their development can be divided among multiple teams, and they can use whatever language they require or from which they can benefit the most this way more performant services can be developed increasing the overall efficiency of the application. The application communicates to one another through API[4], so if the other application conforms to the same API, the services developed can prove to be reusable.

With the development of microservices industry has also grown, now tools are specifically built to support this architecture, these tools can be categorized under various domain, Container engine[5], Service discovery[6], monitoring[7], container occustraction[8], fault tolerance[9], Continuous delivery system using these technologies a full-scale microservice-based application is developed which are highly scalable, flexible and maintainable. By using this approach the application to build can be broken into smaller components that can be built and maintained by a small team. Developing applications is also very economic for the companies, as it has been studied that using this approach reduces the infrastructure cost by 70% or more. That's why major companies like Facebook, Netflix and linked in have adopted this architecture to deploy their application on the cloud as a set of services that are developed, tested deployed and scaled independently of one another while maintaining compatibility with other services.

II. BACKGROUND

Prior to the advent of 3G network technology the traffic on the internet was not high, and there was very small traffic from the mobile devices neither the applications were of the scale they are now nor the web application has so many capabilities and nor they have to deal with such huge demands. So, when 3G became the de facto standard from mobile phones a new possibility emerged and came the surge in the number of people who were online all the time. Now the application needed to scale for thousands and millions of users as SaaS [10] products became more common, new

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 6, June 2020 ||

startups came with ambitious and large-scale products, new markets emerged, and the business moved from B2B to B2C approach.

The problem which came with this is that at this time application were mainly built on, monolithic architecture that means the entire application was one and had a common code base, under this methodology the development was slow and maintains was even more complicated as to make modification entire application was need to brought down and the deployed again, apart from the maintenance issue it was also very expensive to work with the monolithic application because scaling[] them entirely new instance of the application is needed to be launched even if only some components of the application are only required. For few company's SOA [11] came as a remedy, the application was divided into services that complement each other, but these solutions were only sufficient for enterprise-level applications, and were not suitable for internet level applications, which had to deal with millions of peoples.

SOA laid the foundation for the microservice architecture and companies who were into consumer market like Amazon, Netflix, Google and Facebook the giants adopted this method of development to fulfill their ever-growing demand, these companies poured in millions of dollars into the research to develop new tools and technology to support microservice-based development approach. the microservice design pattern [12] it advised that an application can be broken into a set of services, which can be developed independently thus separating the code base of the service. And to communicate between these services REST [13] (Representational State transfer) was adopted as it was an agreed standard among the industries and was very compatible with the HTTP [14] on the internet and at the persistence layer.

In Front of this service sits the gateways whose responsibility was to get requests from the various types of users that can either be Android devices, IOS devices or IoT devices. These gateways used to expose these services using different endpoints. Now since the application was developed using an independent team and these services communicated using endpoints, these services can now be completely decoupled and scaled and have their separate codebase. This brought a great opportunity for the developers now they can take advantage of other languages and take advantage of features each language provides.

Along this the cloud infrastructure was also booming and rapid development was happening in this space huge investments were made for developing world-class infrastructure to build a cloud platform, amazon and google invested built huge data farms for dealing with the traffic they were getting and started renting out their infrastructures, these brought development of container technology docker[15] which started as an open-source project. This improved who applications were built, deployed and scaled other tools came into play for managing these containers, like Kubernetes [16] a container occultation tool to manage the containers.

The microservice architecture is not only efficient in terms of performance, but this approach is also highly cost-efficient reducing the cost of operation to huge proportions, as it is now no longer required that the application are up all the time only necessary services are needed to be kept online and rest can be spun-up as when required and now instead of scaling the whole application only the services which have greater demand are scaled up and when the request is not there they are brought down

III. TOOLS AND TECHNIQUES

Container Engine a container is unit of that packs of the codebase and their dependencies under in a single system that can be deployed independently of the operating system, it provides a level of isolation to run the application and run multiple instances of the same, the container requires an operating system and is set up above it which provide service to the containerized application, the advantage of using a containers, is that they have a lower overhead than running the application than on a virtual machine [17], and since the container are run over an operating system, it can is much easier to communicate among the containers. One of the popular tools in the current industry is docker, which is based on linux libcontainer.

Service discovery when the application is developed it is not possible to know where the application will be deployed as they are deployed on the cloud-based system, they are spun up, reallocated, replicated, and taken down which make tracking them a little complex process. To manage a microservice architecture a service discovery mechanism is deployed, which manages a service registry, similar to SOA. The registry is the service that is used by other components to retrieve the information about the other components, microservices also publish their information back to the registry service, the client also uses the registry to discover the related service. There are many custom implementations of service registry in microservice architecture like Netflix Eureka [18], AWS Elastic load balancing [19].

Monitoring is a critical part of running a micro-structure-based application. monitoring a microservice application is more complex as the are many components that can fail, performance is a nonbinary term, so it is also important to monitor the detailed performance of each of these components, and since most of these will be deployed on cloud

infrastructure for which the user has to enter in service level argument, and without monitoring it is not possible to know if the SLA is being honored by the service provider. For microservice-based applications it is necessary to monitor the application matric, platform metric, and system metric. Tools that are used to monitor are Apache Zepkin, Grafana, Prometheus.

Container Occustraction, with the feature like auto-scale and advent of containers like docker, it is difficult for a human to manage a large set of containers, so container occultation is required to select, to deploy, to monitor and dynamically control and configure the containers that are deployed on the cloud, occsustraction is not only necessary for the deployment of the application but also manage their runtime, and to all that as less human interactions as possible. A major tool for container occultation is Kubernetes [20], which helps to deploy containers in complex cloud environments.

Fault tolerance, though microservice-based applications are highly scalable still there is an upper limit to which capacity and that is due to the fact the scaling the application more than that can be very expensive or some resource has been taken down for maintenance, this can cause a cascading effect on the other service, one of the service breaker implementations is Hystrix which stops cascading problems, provides fallbacks and graceful degradation, and manages rapid recovery.

IV. CONCLUSION AND FUTURE WORK

Throughout this report we reviewed emergence on microservice architecture and how it has become a defacto standard for building highly scalable applications, which are able to handle ever-growing internet traffic and this architecture needs to develop much more and to handle the new surge that will come after the use of 5G[21] begins. We also looked at the technology and tools which help in building and deploying these services: container engine, service discovery, monitoring, container occultation and fault tolerance. These tools and patterns have been developed and being improved to perform better and give a better experience to the users.

V. ACKNOWLEDGMENT

We are thankful to Dr. K N Subramanya, Principal, R V College of Engineering® for his encouragement and valuable guidance. We also thank the Department of Computer Applications (MCA), RV College of Engineering®, Bengaluru for their support.

REFERENCES

1. H. Ekstrom et al., "Technical solutions for the 3G long-term evolution," in IEEE Communications Magazine, vol. 44, no. 3, pp. 38-45, March 2006, doi: 10.1109/MCOM.2006.1607864.
2. Bondi, André B. "Characteristics of scalability and their impact on performance." *Proceedings of the 2nd international workshop on Software and performance*. 2000.
3. Erl, Thomas. *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 1900.
4. Chen, Mingtse, Anil K. Annadata, and Leon Chan. "Adaptive communication application programming interface." U.S. Patent No. 7,581,230. 25 Aug. 2009.
5. Stubbs, Joe, Walter Moreira, and Rion Dooley. "Distributed systems of microservices using docker and serfnode." *2015 7th InternationalWorkshop on Science Gateways*. IEEE, 2015.
6. Czerwinski, Steven E., et al. "An architecture for a secure service discovery service." *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. 1999.
7. Saman, Barakat. "Monitoring and analysis of microservices performance." *Journal of Computer Science and Control Systems* 10.1 (2017): 19.
8. Khan, "Key Characteristics of a Container Orchestration Platform to Enable a Modern Application," in IEEE Cloud Computing, vol. 4, no. 5,
9. Montesi, Fabrizio & Weber, Janine. (2016). Circuit Breakers, Discovery, and API Gateways in Microservices.
10. Gold, Nicolas, et al. "Understanding service-oriented software." *IEEE software* 21.2 (2004): 71-77.
11. Niknejad, Naghmeh, et al. "Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation." *Information Systems* (2020): 101491.

| e-ISSN: 2319-8753, p-ISSN: 2320-6710| www.ijirset.com | Impact Factor: 7.512|

|| Volume 9, Issue 6, June 2020 ||

12. Hodges, Jason Lee. "Design Patterns." *Software Engineering from Scratch*. Apress, Berkeley, CA, 2019. 293-304.
13. Zhou, Wei, et al. "REST API design patterns for SDN northbound API." *2014 28th international conference on advanced informationnetworking and applications workshops*. IEEE, 2014.
14. Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." *Linux journal* 2014.239 (2014): 2.
15. Burns, Brendan, et al. "Borg, omega, and kubernetes." *Queue* 14.1 (2016): 70-93.
16. Canturk, Isci, et al. "Virtual machine demand estimation." U.S. Patent No. 9,037,717. 19 May 2015.
17. Leung, Andrew, Andrew Spyker, and Tim Bozarth. "Titus: introducing containers to the Netflix cloud." *Queue* 15.5 (2017): 53-77.
18. Ferris, James Michael. "Load balancing in cloud-based networks." U.S. Patent No. 8,849,971. 30 Sep. 2014.
19. Brewer, Eric A. "Kubernetes and the path to cloud native." *Proceedings of the Sixth ACM Symposium on Cloud Computing*. 2015.
20. Andrews, Jeffrey G., et al. "What will 5G be?." *IEEE Journal on selected areas in communications* 32.6 (2014): 1065-1082.
21. M. Villamizar et al., "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, 2016, pp. 179-182, doi: 10.1109/CCGrid.2016.37.
22. P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis and S. Tilkov, "Microservices: The Journey So Far and Challenges Ahead," in IEEE Software, .vol. 35, no. 3, pp. 24-35, May/June 2018, doi: 10.1109/MS.2018.2141039.
23. M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," 2015 10th Computing Colombian Conference (10CCC), Bogota, 2015, pp. 583-590, doi: 10.1109/ColumbianCC.2015.7333476.
24. N. Alshuqayran, N. Ali and R. Evans, "A Systematic Mapping Study in Microservice Architecture," 2016 IEEE 9th International Conferenceon Service-Oriented Computing and Applications (SOCA), Macau, 2016, pp. 44-51, doi: 10.1109/SOCA.2016.15. pp. 42-48,September/October 2017, doi: 10.1109/MCC.2017.4250933.