Identification of the deployment defects in Microservice hosted in advanced VCS and deployed on containerized cloud environment

Amarjeet Singh
710 Duncan Ave Apt#1409 Pittsburgh, Pennsylvania, 15237, USA amarteotia@gmail.com

Vinay Singh
6032 Blue Ridge Dr, Apt#A, Highlands Ranch, Colorado, 80130, USA
vsbuild7@gmail.com

Alok Aggarwal
School of Computer Science, University of Petroleum & Energy Studies Dehradun, India
alok.aggarwal@ddn.upes.ac.in

Shalini Aggarwal
Uttaranchal Intt. of Mgt., Uttaranchal University, Dehradun, Shalinia289@yahoo.com

Pratibha Pandey
Uttaranchal Intt. of Mgt., Uttaranchal University, Dehradun,
pratibhapandey8502@gmail.com

Manisha Khanduja
Uttaranchal Intt. Of Mgt., Uttaranchal University, Dehradun,
manisha.khanduja@gmail.com

Abstract.

Micro-services architecture has evolved as the popular software development model for the enterprise applications. Since enterprise applications are complex by nature and they require out of the box scalability and low latency, these hence Micro-services provides a significant contribution in accomplishing these objectives. Enterprises can achieve a long-term vision of an API-enabled, loosely coupled, highly scalable and flexible platform architecture with Micro-services in containerized cloud environment. It has been observed that there are very few works available on this topic. Software industry needs to emphasize in the area to find out the defects at initial level rather than seen errors in production environments. Few works focus on Micro-services while others on Kubernetes issues and challenges but there is no relation between the two has been found. That motivated us to go for the proposed work, which depicts to identify the defects in early stage for Micro-services deployed on Kubernetes. Few standard basic guidelines for the micro-service architecture, in terms of naming convention, automation, monitoring & warning, fault design, and design philosophy, are proposed.

Key words. Version Control System, Git, Subversion, micro-service, Kubernetes, container

I. INTRODUCTION

As companies embark on digital transformation, the enterprise architecture team aims to design a flexible and scalable architecture. The most important drivers for micro-service architecture are, Duplicate the System of Engagement (SoE) and the System of Records (SOR) using clearly defined API agreements. It provides flexibility to change tools and technologies without affecting experience and business performance. Build large, complex, high-fault tolerance systems that offer scalability and availability to meet the needs, use the lean model to create presentation and integration layers to create responsive and high-performance experiences. The platform should offer needs-based scalability and high availability. Adopt DOOPS principle for continuous integration, continuous delivery with faster release times. Each micro service has its own data model and manages its own data. Data is transferred between micro-services using "mute pipes" like light protocols such as Event Broker and / or REST. Smaller in scope that encompasses the same business functionality. The internal operation is a "black box" from which external programs can only access through the API [1][2]. The foremost common issues are the weakening of the picture quality of the container and attempting to use special pictures without indicating the registration data. Typically, particularly difficult if we are starting to work with Kubernetes or utilizing CI/CD for the primary time [3]. Kubernetes informational suggest that you simply transmit the arrangement using the setup outline or in mystery when the application begins. This information may incorporate database qualifications, API endpoints, or other setup banners [4]. A common botch of engineers is to make applications that don't have and don't have the reference properties of setup maps or arrangement maps/secrets. Whether we are propelling a new app on Cubernets or moving to an existing stage, apps frequently crash early. We have diverse situations and we have a partitioned cabinet set for each environment; one improvement set and two generation sets. We have our possess computer program within the advancement cluster (isolated application) and we utilize our claim computer program (single application, smaller scale administrations, Crown, etc.) within the advancement cluster. We as of now have around ten organizations per diminutive and thousands of module arrangements. With the growing number of holders within the cluster, we are confronting the issue of moderate arranging [5]. For this issue to happen, 70 conditions on a single hub were adequate. It took a few minutes in a push. Moderate planning is no issue in a generation environment where we are dealing with heavy workloads and CPU utilization and thus we don't require more than 70 modules per hub. Kubernetes offers two primary functions, to begin with first is called life sensors and second one is accessibility sensors. Fundamentally, the lifetime/uptime sensor will perform a few activity from time to time to confirm that our application is working accurately (such as sending an HTTP ask, opening a TCP association, or a command in your holder) To act). We confront a number of arrange issues such as tall delays and parcel dropping. In any case, the root of the issue is the Linux part, not the Cybernet itself. The execution of cybernetics largely depends on the execution of the Linux bit. You ought to unquestionably check the processor recurrence control mode. We

ran LXC sometime recently Kubernetes and a few of its administrations moved to Kubernetes without changing the code base. A curiously thing was that their execution on the K8s was second rate to that of the LXC. Indeed the foremost IT divisions appeared moo execution in utilizing the same CPU. On the off chance that the life test comes up short, it will crush our holder and make a modern one. On the off chance that the status test comes up short, this module cannot be gotten to as a benefit endpoint, which implies that activity will not be sent to this module until it is prepared. Within the current conveyance, Progressed Nginx servers point to Kubernetes endpoints by default. The logon controller listens to occasions and resets the grouping without interference. However, let's take a closer see at the method of steady upgrading. Sends and reports any startup prepare to the holder related with the Kubernetes module. At that point report it to the Kubernetes API. Since these forms run in parallel, a few administrations may as of now be down, but will proceed to be sent to the activity module when entering from the current upstream [6]-[8]. Figure 1 shows the Kubernetes eco system for Micro-services.

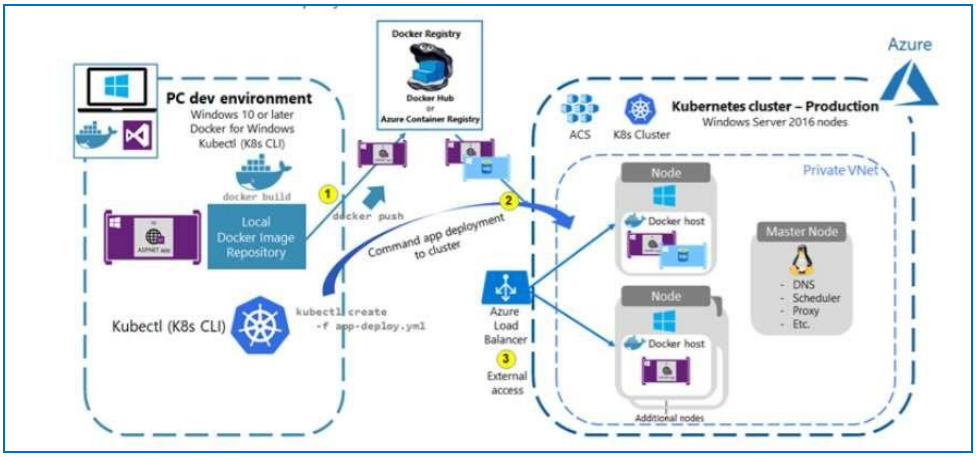

Figure 1. Kubernetes eco system for Micro-services

Few works focus on Micro-services while others on Kubernetes issues and challenges but there is no relation between the two has been found. That motivated us to go for the proposed work, which depicts to identify the defects in early stage for Micro-services deployed on Kubernetes [9-14]. Few standard basic guidelines for the micro-service architecture, in terms of naming convention, automation, monitoring & warning, fault design, and design philosophy, are proposed. Rest of the paper is organized as follows. Section 2 gives the proposed methodology. Results and discussion are given in section 3. have found it difficult and has been interpreted as per the project requirement. Actual granularity for a micro-servicestrategic distance [15] from this, make unchanging holders. In case of any surrenders or vulnerabilities, designers can revamp and redeploy holders.

Inaccessible administration is done through runtime APIs or by making farther shell sessions to the have on which the Micro-services are running [16].

II. METHODOLOGY

The word “micro” used in Micro-services world architects adequate. It took a few minutes in a push. Moderate planning is no issue in a generation environment where we are dealing with heavy workloads and CPU utilization and thus we don't require more than 70 modules per hub. Kubernetes offers two primary functions, to begin with first is called life sensors and second one is accessibility sensors. Fundamentally, the lifetime/uptime sensor will perform a few activity from time to time to confirm that our application is working accurately (such as sending an HTTP ask, opening a TCP association, or a command in your holder) To act). We confront a number of arrange issues such as tall delays and parcel dropping. In any case, the root of the issue is the Linux part, not the Cybernet itself. The execution of cybernetics largely depends on the execution of the Linux bit. You ought to unquestionably check the processor recurrence control mode. We ran LXC sometime recently Kubernetes and a few of its administrations moved to Kubernetes without changing the code base. A curiously thing was that their execution on the K8s was second rate to that of the LXC. Indeed the foremost IT divisions appeared moo execution in utilizing the same CPU. On the off chance that the life test comes up short, it will crush our holder and make a modern one. On the off chance that the status test comes up short, this module cannot be gotten to as a benefit endpoint, which implies that activity will not be sent to this module until it is prepared. Within the current conveyance, Progressed Nginx servers point to Kubernetes endpoints by default. The logon controller listens to occasions and resets the grouping without interference. However, let's take a closer see at the Rest of the paper is organized as follows. Section 2 gives the proposed methodology. Results and discussion are given in section 3. have found it difficult and has been interpreted as per the project requirement. Actual granularity for a micro-service component is required to be de-scoped. Therefore, the actual focus area is to improve the reusability for a micro-service component in a functional domain. Software project teams devoting their efforts to create the Micro-services are presented with many challenges. Close to 70 percent of the project teams require the refactoring and analyzing of the source code as a major factor in the development [9]-[16]. Designers tend to take off shell get to images so they can settle them in generation. Be that as it may, assailants regularly misuse this get to infuse malevolent code. To maintain a strategic distance from this, make unchanging holders. In case of any surrenders or vulnerabilities, designers can revamp and redeploy holders. Inaccessible administration is done through runtime APIs or by making farther shell sessions to the have on which the Micro-services are running[17-20].

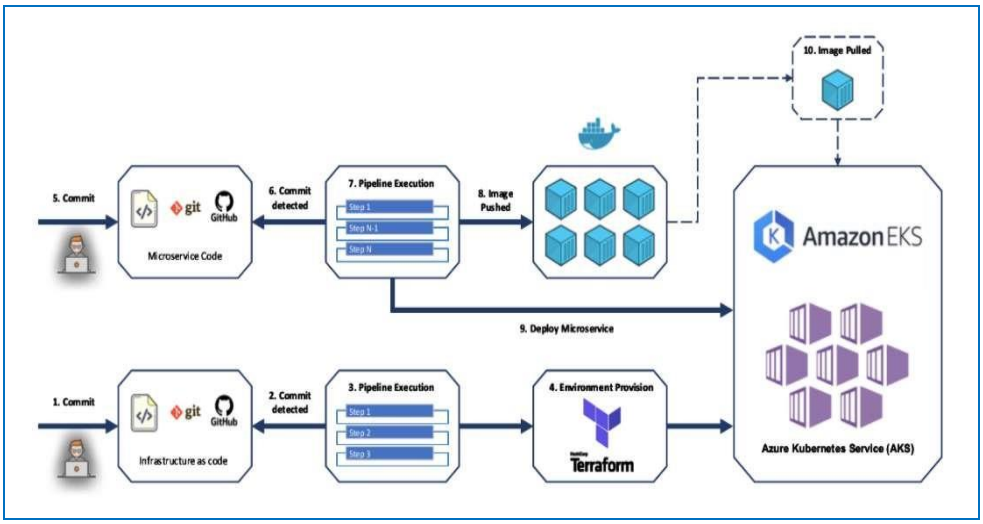

Figure 2. Kubernetes deployment Architecture over Amazon cloud

Therefore, there are numerous open-source bundles for designers with promptly accessible holders, counting Node.js, Apache Web Server and the Linux working framework. In any case, for security purposes, we would like to know where holders start, when they were upgraded, and in the event that they're free of any known vulnerabilities and malicious code. It also included the term of aggregates in interface [21].

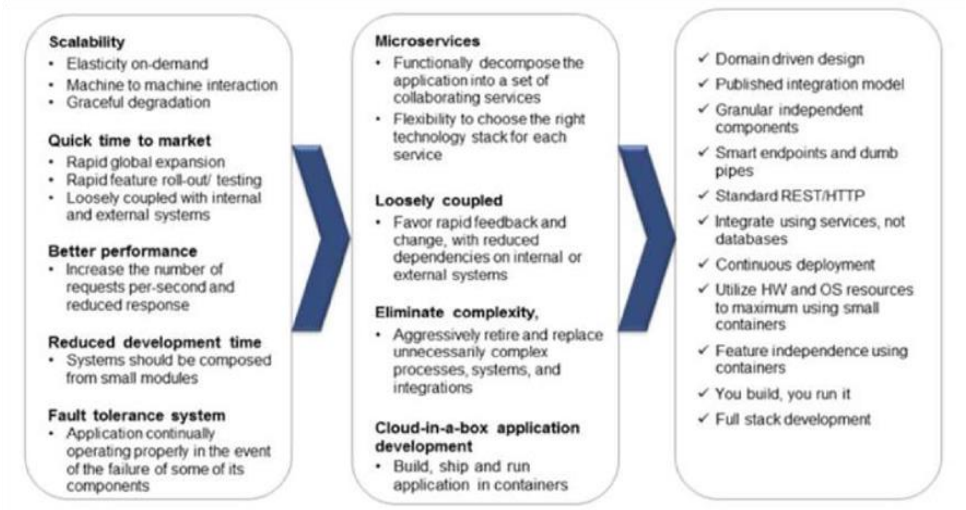

Figure 3. The challenges and Features of Micro-services

It’s best to set up a trusted picture store and run pictures as it were from that trusted source. Kubernetes deployment Architecture over Amazon cloud is shown in figure 2 and figure 3 shows the challenges and Features of Microservices.

III. RESULTS AND DISCUSSION

One of the most popular and important plan methodologies for designing cloud-native frameworks is micro-service engineering. We are confident that the software industry clients will embrace this trend in the integration of distributed frameworks. Regardless, the problems and obstacles tend to lead to sub-optimal implementations of this architectural style, leading to an atmosphere in which organisations continue to believe that Micro-services is just another adopted fad. The additional complexity of operating and troubleshooting these frameworks creates new problems. Wherever any of these hazards or challenges lead to strategic decisions, we must ensure that such unique situations are handled by a specialist debt-management mechanism. Finally, the maturity of the framework determines how well ventures can solve these difficulties. The following are basic micro-service architecture naming conventions, automation, monitoring and warning, fault design, and design philosophy standards. Naming conventions: The URL of Micro-services is usually a name that represents the source. We will use it for the program perform the right actions; example Get api/v1/accounts that will list all accounts. Automation: In order to reduce the operational complexity of the microservice architecture, we need to automate the operational complexity tasks such as compiling, deploying, error reporting, alerts, monitoring, automatic scaling, and others. Monitoring and Warning: Tools should be used to monitor the performance and availability of Micro-services watch.

October monitoring services can be configured to monitor disk space, CPU usage, and other settings. To alert operational teams in case of violation of the service level agreement, we need to configure the appropriate thresholds. Fault design: In order to handle faults, we need to implement functions such as Version Control - Micro-service versions are managed using versions that are part of the micro-service endpoint. Design philosophy: Detailing of Micro-services should be based on the following principles; Business functionality (Each micro-service must be designed to show a single business functionality), DevOps Installation (We need to configure the DevOps ecosystem according to the structure and distribution pipeline, Management (We need to define the standards of functionality/performance improvement, implementation and verification). Distributed design: Since the system consists of several Micro-services, we must have the most suitable one parsing of services, clean interfaces for services and convenient database for each service. Finally, the strive of software companies to solve these problems depends on the maturity of the infrastructure and the competence and consistency of design in it business and IT in general. Try to develop before that instead of trying to overcome all this underwater, the states gems and challenges ahead of us, learn and customize our own forward.

REFERENCES

[1] Hou Q., Ma Y., Chen J., and Xu Y., 'An Empirical Study on Inter-Commit Times in SVN,' Int. Conf. on Software Eng. and Knowledge Eng.," pp. 132–137, 2014.

[2] O. Arafat, and D. Riehle, 'The Commit Size Distribution of Open Source Software,' Proc. the 42nd Hawaii Int'l Conf. Syst. Sci. (HICSS'09), USA, pp. 1-8, 2009.

[3] C. Kolassa, D. Riehle, and M. Salim, 'A Model of the Commit Size Distribution of Open Source,' Proc. the 39th Int'l Conf. Current Trends in Theory and Practice of Comput. Sci. (SOFSEM'13), Czech Republic, pp. 52–66, 2013.

[4] L. Hattori and M. Lanza, 'On the nature of commits,' Proc. the 4th Int'l ERCIM Wksp. Softw. Evol. and Evolvability (EVOL'08), Italy, pp. 63–71, 2008.

[5] P. Hofmann, and D. Riehle, 'Estimating Commit Sizes Efficiently,' Proc. the 5th IFIP WG 2.13 Int'l Conf. Open Source Systems (OSS'09), Sweden, pp. 105–115, 2009.

[6] Kolassa C., Riehle, D., and Salim M., 'A Model of the Commit Size Distribution of Open Source,' Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'13), Springer-Verlag, Heidelberg, Baden-Württemberg, p. 5266, Jan. 26-31, 2013.

[7] Arafat O., and Riehle D., 'The Commit Size Distribution of Open Source Software,' Proceedings of the 42nd Hawaii International Conference on Systems Science (HICSS'09), IEEE Computer Society Press, New York, NY, pp. 1-8, Jan. 5-8, 2009.

[8] R. Purushothaman, and D.E. Perry, Toward Understanding the Rhetoric of Small Source Code Changes, IEEE Transactions on Software Engineering, vol. 31, no. 6, pp. 511–526, 2005.

[9] A. Alali, H. Kagdi, and J. Maletic, 'What's a Typical Commit? A Characterization of Open Source Software Repositories,' Proc. the 16th IEEE Int'l Conf. Program Comprehension (ICPC'08), Netherlands, pp. 182-191, 2008.

[10] A. Hindle, D. Germán, and R. Holt, 'What do large commits tell us?: a taxonomical study of large commits,' Proc. the 5th Int'l Working Conf. Mining Softw. Repos. (MSR'08), Germany, pp. 99-108, 2008.

[11] Alok Aggarwal, Vinay Singh, Narendra Kumar, 'A Rapid Transition from Subversion to Git: Time, Space, Branching, Merging, Offline Commits & Offline builds and Repository Aspects,' Recent Advances in Computer Science and Communications, vol. 14: e210621194190, 2021.

[12] V. Singh, M. Alshehri, A. Aggarwal, O. Alfarraj, P. Sharma et al., 'A holistic, proactive and novel approach for pre, during and post migration validation from subversion to git,' Computers, Materials & Continua, vol. 66, no.3, pp. 2359–2371, 2021.

[13] Vinay Singh, Alok Aggarwal, Narendra Kumar, A. K. Saini, 'A Novel Approach for Pre-Validation, Auto Resiliency & Alert Notification for SVN To Git Migration Using Iot Devices,'PalArch's Journal of Arch. of Egypt/Egyptology, vol. 17 no. 9, pp. 7131 – 7145, 2021.

[14] Singh Vinay, and Aggarwal Alok, 'Performance Analysis of Middleware Distributed and Clustered Systems (PAMS) Concept in Mobile Communication Devices Using Android Operating System,' Proc. Third IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC-2014), December 11-13, 2014, Waknaghat, Solan, India.

[15] V. Singh, A. Singh, A. Aggarwal and S. Aggarwal, 'A digital Transformation Approach for Event Driven Micro-services Architecture residing within Advanced VCS,' 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), 2021, pp. 100-105, doi: 10.1109/CENTCON52345.2021.9687973.

[16] V. Singh, A. Singh, A. Aggarwal and S. Aggarwal, 'DevOps based migration aspects from Legacy Version Control System to Advanced Distributed VCS for deploying Micro-services,' 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), 2021, pp. 1-5, doi: 10.1109/CSITSS54238.2021.968371

[17] Kumar, A., Memoria, M., & Kumar, V. (2021). Memory optimized deep learning based face recognization. Indian Journal of Computer Science and Engineering, 12(1), 57–64. https://doi.org/10.21817/indjcse/2021/v12i1/211201066

[18] Kumar, A., Singh, D., & Punia, P. (2016). Implementation of image dehazing technique using image fusion. International Journal of Control Theory and Applications, 9(20), 307–315. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85006371817&partnerID=40&md5=17ff92c722005ff83c9c27c371a0f2ed

[19] Kumar, A., & Verma, A. (2016). Missing numbers in graceful graphs. International Journal of Control Theory and Applications, 9(21), 133–136. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85008259815&partnerID=40&md5=b85175173bb98fecc1b0ec98e2185b98

[20] Kumar, D., Chibber, V. K., & Singh, A. (2018). Physical and chemical properties of Mahua and Sal seed oils. In Advances in Intelligent Systems and Computing (Vol. 624, pp. 1391–1400). https://doi.org/10.1007/978-981-10-5903-2_146

[21] Kumar, M., Chandramauli, A., & Ashutosh. (2018). Partial replacement of fine aggregates of fire bricks with fine aggregates in concrete. International Journal of Civil Engineering and Technology, 9(3), 961–968. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85045069805&partnerID=40&md5=65b0161451cfa3352689f31d23a5b191

Biographies

Amarjeet Singh earned a Master's degree in Computer Science from Southern Arkansas University in Arkansas USA in 2021 and Bachelor's degree in Information Technology in 2005. Mr. Amarjeet Singh possesses over 15 years of experience in the Software and IT industry in multiple roles in design development and architecture of modern cloud and Microservices based systems. Provided help to many clients for the digital transformation journey.

Vinay Singh has got 17 years of experience in different domains and technologies in the Software Industry and Teaching. He has worked with AT&T, CapitalOne Bank, Agilent, Walgreens kind of industry leading organization. He is currently working as Manager Software Engineer with Charles Schwab, USA. With this Progressive Experience as leader to DevOps and Software engineering, he is leading a big team of engineer to Build & Release with Continuous Integration/Continuous Deployment using Jenkins, Bamboo and Maven under Agile framework from Dev to Production.

Alok Aggarwal earned PhD degree in Mobile Computing area from IIT Roorkee, INDIA in 2010, Master's degree in Computer Science & Engineering in 2001 and Bachelor's degree in Computer Science in 1995. Contributing over 23+ years in Teaching (CSE & IT), as well as S/W Development. Currently spearheading efforts as Professor (CSE), with

University of Petroleum & Energy Studies, Dehradun (UK) INDIA. He has contributed 250 research papers, 5 Patents, 5 Books and 4 Book Chapters. Current research interests include Machine learning, AI, power control management, MANET & wireless sensor networks.

Shalini Aggarwal earned PhD degree in networking in 2018 from Mewar University Rajasthan, Master's degree in Computer Applications in 2007 and Bachelor's degree in Science in 2001. Contributing over 12^{+} years in Teaching and currently working as Assistant Professor in Uttaranchal University. She has contributed 12 research papers.

Pratibha Pandeyis working as Assistant Professor at Uttaranchal University, Dehradun, India. She is MCA from Punjab Technical University, Jalandhar. M.Tech (Computer Science& Engineering), from Uttarakhand Technical University, Dehradun, Uttarakhand.Her area of interest includes Machine Learning, Big Data, Cyber Security, and Cloud Computing.

Manisha Khandujais working asAssistant Professor at Uttaranchal University, Dehradun, India. She is MCA fromUttarakhand Technical University, Dehradun, Uttarakhand.Her area of interest includes Data Mining, Internet of Things etc.