A SELF HEALING SOFTWARE SYSTEM USING DEVOPS AND AUTOMATION

Dr. S. B. Chaudhari[1], Md. Arslaan Thanedar[2], Gaurav Agrawal[3] , Faizan Quadri[4]

Dept. of Computer Engineering,
J.S.P.M's Jayawantrao Sawant College of Engineering, Pune, India.

---***---

Abstract - The increasing complexity of modern software systems and the demand for high availability and reliability have necessitated the development of self-healing software systems. This paper proposes a comprehensive architecture for developing such systems utilizing automation and DevOps practices. Specifically tailored for cloud-based environments, the proposed architecture leverages continuous integration and continuous deployment (CI/CD) with Jenkins, containerization with Docker, and automated bash scripts for self-healing. Additionally, the architecture includes tools for monitoring to provide automated metrics of the software in runtime. At the container level, self-healing will be implemented using Kubernetes. The aim of this project is to create a self-healing software deployment system that can detect and recover from faults with minimal human intervention, thereby enhancing the reliability, resilience, and efficiency of software systems.

Key Words: DevOps, Automation, Containerization, Self Healing software systems.

1. INTRODUCTION

In today's fast-paced digital landscape, where software applications are the lifeblood of businesses and organizations, ensuring uninterrupted service and rapid issue resolution is of paramount importance, the proposed software system using devOps[15] and automation represents an endeavor in the realm of software application deployment and system reliability.

Traditional software systems are often fragile and can fail catastrophically if a single component fails. Self-healing systems, on the other hand, are designed to be resilient and can continue to operate even if some of their components fail. Self-healing is the ability of a system to return to a functional state after being compromised by an anomalous agent. Self healing[3] systems endeavor to heal themselves in a similar fashion as a biological system heals from a wound from faults and recovers into normalcy. Self-healing systems are particularly well-suited for DevOps environments[7], where software is constantly being developed, deployed, and tested. In these environments, self-healing systems can help to reduce the burden on DevOps teams, allowing them to focus on more strategic tasks. Automation is another key component of self-healing systems. Automation tools can be used to monitor systems for failures, execute remediation actions, and learn from past failures to improve the system's ability to heal itself.

2. LITERATURE SURVEY

The analysis of similar papers for this report reveals a variety of effective approaches to creating self-healing software systems. These approaches range from using traditional approaches to predict and prevent faults to implementing automated recovery mechanisms. However, while these approaches are effective, they often lack a comprehensive system architecture that integrates all the necessary components for a truly self-healing system.

Title	**Authors**	**Year**	**Remarks**
Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review [1]	F. Khan, M. Khan, and T. Whangbo	2022	This study offers an in-depth exploration of the challenges associated with implementing DevOps culture and practices. However, there are several areas where it can be improved upon. For example, the paper could benefit from a more detailed analysis of the specific tools and practices that can help address these challenges.
Evaluation of Self-Healing Systems: An Analysis of the State-of-the-Art And Required Improvements. [2]	Sona Ghahremani, Holger Giese	2020	This study is focused on Self-Adaptive Systems (SAS) and Self-Healing Systems (SHS) based on traditional servers. However, it lacked a sufficient evaluation of errors and their remedies. It highlights the need for improvement in the current self-healing approaches.

Framework to Deploy Containers using Kubernetes and CI/CD Pipeline.[3]	Manish Kumar Abhishek, D. Rajeswara Rao, K. Subrahmanyam	2022	This work is centered around the utilization of Kubernetes and Continuous Integration/Continuous Deployment (CI/CD) pipelines for deploying containers. The study identifies a limitation related to pods synchronization in clusters, indicating a requirement for enhanced cluster management.
Self-Healing Systems: Application and Methodologies - A Review.[4]	Anigbogu S.O, Anigbogu K.S	2020	This review is primarily based on self-healing approaches. While it provides valuable insights, it identifies a dearth of comprehensive information on errors and their corresponding remedies, and also highlights a lack of self-observation approaches in the current methodologies.
Evaluating Software Automation Enhancement through the Implementation of DevOps: A Comparative Analysis and Future Directions[5]	Sarthak Srivastava, Karthik Allam, Anirudh Mustyala	June 2023	This study focuses on the advancements in software automation within the context of DevOps implementation. Drawing parallels with the state-of-the-art in self-healing systems (SAS and SHS) based on traditional servers.

3. OBJECTIVES

The objective of a self-healing software system for a server is to automatically detect and remediate software failures[10], ensuring that the server remains up and running even in the event of a failure. This can be achieved by using a variety of automation tools and techniques

1. **Deploy:** Deploy the user application by single step automated deployment

2. **Monitor:** Employ monitoring techniques to track system components and interactions, enabling early detection of anomalies and ensuring optimal system functioning

3. **Detect:** Continuously assess system health and performance metrics to identify anomalies and potential issues in real-time.

4. **Heal:** Employ automated processes to swiftly detect and rectify system disruptions, minimizing downtime and ensuring seamless operation
.

5. **Report:** Generate comprehensive and actionable reports detailing system incidents, resolutions, and performance trends for informed decision-making

The objectives of deploying, monitoring, detecting, healing, and reporting within an application serve critical roles in ensuring the efficiency, reliability, and stability of system operations. Deploying applications[3] via a single-step automated process streamlines deployment workflows, saving time and reducing errors. Monitoring enables real-time tracking of system components, preemptively identifying anomalies for prompt resolution, hence ensuring optimal system performance. Continuous detection of system health and performance metrics facilitates proactive troubleshooting, while automated healing processes swiftly address disruptions, minimizing downtime. Finally, comprehensive reporting provides actionable insights into system incidents and trends, empowering informed decision-making[15] for ongoing system optimization and enhancement. Together, these objectives fortify the application's resilience and effectiveness in dynamic operational environments[11].

4. PROPOSED ARCHITECTURE

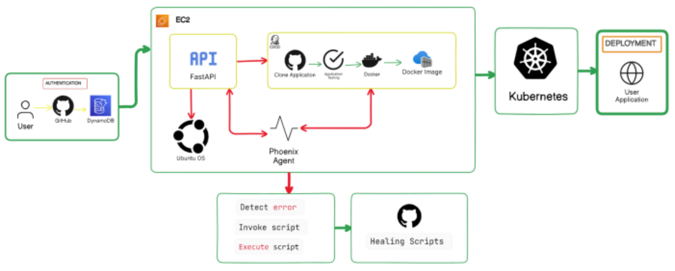

Fig: Proposed Architecture of A Self Healing System Using Automation

The above diagram illustrates a versatile architecture designed for developing a self-healing and monitoring automation application[3] tailored for cloud[7] deployment scenarios. This application streamlines the process of deploying an application to the cloud. Initially, users[9] are prompted to authenticate via their GitHub account and select a repository for deployment. Subsequently, the chosen repository undergoes cloning and dockerization, culminating in the generation of a Docker image within a Jenkins[12] server environment. This entire workflow unfolds within an Amazon EC2 instance[17], housing a Python agent responsible for event log monitoring. In the event of an error, the Python agent promptly intervenes, executing pre-configured scripts to rectify the issue and restore system functionality. The self healing[3] ability of the containers can be realized using kubernetes[15]. The overarching goal is to establish a self-sufficient system that autonomously addresses errors while maintaining continuous monitoring, thereby minimizing the need for human intervention..

5. CONCLUSIONS

In conclusion, the proposed architecture for a self-healing software system using DevOps and automation offers a comprehensive and automated approach to creating resilient and reliable software. By leveraging continuous integration and continuous deployment with Jenkins, infrastructure as code with Terraform, containerization[13] with Docker, and automated bash scripts for self-healing, organizations can create software that is able to detect and recover from faults with minimal human intervention. Additionally, the use of GitHub for version control, Kubernetes for container orchestration[14], and Amazon RDS for relational database management ensures that the software is scalable, reliable, and secure.

Looking to the future, there is significant potential for further development and improvement of this architecture. For example, the use of machine learning algorithms could enhance the self-healing capabilities of the software by predicting and preventing faults before they occur. Additionally, the integration of artificial intelligence and natural language processing could improve collaboration and communication between development and operation teams.

However, there are also limitations to consider. For example, the reliance on automated scripts for self-healing may not be sufficient for all types of faults, and human intervention may still be required in some cases. Additionally, the use of multiple tools and technologies may introduce complexity and require additional training for development and operation teams.

In summary, while the proposed architecture offers a promising approach to creating self-healing software systems, there is still room for improvement and further research. By addressing the limitations and exploring future developments, organizations can continue to improve the reliability, resilience, and efficiency of their software systems.

REFERENCES

[1] Khan, Muhammad & Khan, Abdul & Khan, Faheem & Khan, Muhammad & Whangbo, Taeg. (2022). Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3145970.

[2] Ghahremani, S. Giese, H. (2020). ”Evaluation of Self-Healing Systems: An analysis of the State-of-the-Art and Required Improvements.”

[3] Abhishek, M. K., Rao, D. R., Subrahmanyam, K. (2022). ”Framework to Deploy Containers using Kubernetes and CI/CD Pipeline.”

[4] Anigbogu, S.O., Anigbogu, K.S. (2020). ”Self-Healing Systems: Application and Methodologies - A Review.”

[5] Tosi, D. (January 2004). ”Research Perspectives in Self-Healing Systems: A Comparative Analysis of Advancements and Future Trajectories.”

[6] Srivastava, S., Allam, K., Mustyala, A. (June 2023). ”Evaluating Software Automation Enhancement through the Implementation of DevOps: A Comparative Analysis and Future Directions.”

[7] Streamlit Documentation. (2024). Retrieved from https://docs.streamlit.io/

[8] Pahl, Claus & Brogi, Antonio & Soldani, Jacopo & Jamshidi, Pooyan. (2017). Cloud Container Technologies: A State-of-the-Art Review. IEEE Transactions on Cloud Computing. PP. 1-1. 10.1109/TCC.2017.2702586.

[9] DynamoDB Documentation. (2024). Retrieved from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/

[10] Gomes J, Bagnaschi E, Campos I, David M, Alves L, Martins J, Pina J, López-García A, Orviz P (2018) Enabling rootless Linux Containers in multi-user environments: the udocker tool. Comput Phys Commun 232:84–97. https://doi.org/10.1016/j.cpc.2018.05.021

[11] Irdin Pekaric, Raffaela Groner, Thomas Witte, Jubril Gbolahan Adigun, Alexander Raschke, Michael Felderer, Matthias Tichy,
A systematic review on security and safety of self-adaptive systems,
Journal of Systems and Software,
Volume 203, 2023,111716, ISSN 0164-1212,
https://doi.org/10.1016/j.jss.2023.111716.

[12]Jenkins Documentation. (2024). Retrieved from https://www.jenkins.io/doc/

[13] Khan, Muhammad & Khan, Abdul & Khan, Faheem & Khan, Muhammad & Whangbo, Taeg. (2022). Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3145970.

[14] Khan, Asif. (2017). Key Characteristics of a Container Orchestration Platform to Enable a Modern Application. IEEE Cloud Computing. 4. 42-48. 10.1109/MCC.2017.4250933.

[15] Kubernetes Documentation. (2024). Retrieved from https://kubernetes.io/docs/

[16] Azeem Akbar, Muhammad & Rafi, Saima & Alsanad, Abeer & Furqan Qadri, Syed & Alsanad, Ahmed & Alothaim, Abdulrahman. (2022). Toward Successful DevOps: A Decision-Making Framework. IEEE Access. 1-1. 10.1109/ACCESS.2022.3174094.

[17] Amazon EC2 Documentation. (2024). Retrieved from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/

[18] N. Zhou, H. Zhou and D. Hoppe, "Containerization for High Performance Computing Systems: Survey and Prospects," in IEEE Transactions on Software Engineering, vol. 49, no. 4, pp. 2722-2740, 1 April 2023, doi: 10.1109/TSE.2022.3229221.