WEB APPLICATION DEPLOYEMENT USING KUBERNETES

L E Sree Sai Praneeth Goud
Department of Computer Science Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India.
bl.en.u4cse21110@bl.students.amrita.edu

N.L.Varshitha
Department of Computer Science Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India.
bl.en.u4cse21128@bl.students.amrita.edu

P. Himanshu
Department of Computer Science and Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India.
bl.en.u4cse21145@bl.students.amrita.edu

Dr. Beena B.M.
Department of Computer Science Engineering
Amrita School of Computing, Bengaluru
Amrita Vishwa Vidyapeetham, India.
bm_beena@blr.amrita.edu

Abstract— With the increasing demand for scalable, highly available, and fault-tolerant web applications, container orchestration platforms like Kubernetes have gained significant popularity. This paper presents a comprehensive approach to deploying web applications using Kubernetes, leveraging its powerful features and capabilities. By containerizing web applications and managing them with Kubernetes, organizations can achieve efficient resource utilization, automated scaling, and seamless deployment workflows. The paper discusses the architecture of Kubernetes, its core components, and the step-by-step process of deploying a web application on a Kubernetes cluster. Additionally, it explores advanced topics such as load balancing, ingress routing, and persistent storage integration, providing insights into best practices and real-world use cases.

Keywords: Containerization, Orchestration, Scalability, High Availability, Automated Deployment.

I. INTRODUCTION

The process of deploying a web application for a Kubernetes environment is a systematic procedure that utilizes Kubernetes features to run and scale the applications properly. Kubernetes is one of the most powerful and flexible container orchestration platforms that conceive and launch instances of applications using the Deployments. These Deployments specify how Kubernetes ought to schedule and run application instances, where replication implies that the instances be spread over several specific Nodes in the cluster, in the most efficient and reliable manner possible.

To begin deploying a Web App in Kubernetes, you require to implement your application in containers and this is mostly done using Docker. Once you package, you use manifests to describe the status of the application and such other details as the number of replicas for a container, the method of update and how the different containers should communicate. When you commit your code to SCM and use a CI/CD platform like Harness, it becomes easy to deploy while at the same time increasing the capacity to achieve it.

Furthermore, one can also expose the application to outside clients by creating services with types LoadBalancer and ExternalName, thus keeping it easily accessible and constant for the general user. To sum up, using Kubernetes requires creating a cluster, putting the web application into containers, defining the application state with manifests making the application available to users, and monitoring it with Kubernetes tools. It offers a sturdy strategy for distributing and utilizing web applications to accommodate the means in a Kubernetes milieu.

II.LITERATURE REVIEW

Issues could possibly emerge in terms of lack of security, network configuration complication, infrastructure management difficulties, no centralized visibility, and resource use limitations might appear while working on Kubernetes as a web application development environment[1]. Kubernetes is a difficult system to analyse due to its complications, which makes it impossible to detect and block illegal access. Managing switching communication among network services, in particular for big-scale applications introduces networking issues as a fundamental

problem. Highly complex infrastructure which comes with the absence of centralized oversight, drives the difficulty of managing the clusters. Eventually, resource utilization may cause the performance to degrade. The implementation of centralized monitoring tools, Kubernetes metrics tracking for efficient resource use, and networking via container network interfaces (CNI) plugins, and a robust security enhancement procedure, using AppArmor and SELinux, are a few solutions of the problems. However, quite a few research gaps exist in order to refine and better the Landing a Kubernetes application web. They include auto configuration of monitoring, sophisticated resource management, upgraded centralized monitoring and logging, security solutions, and advancements in scalability and interoperability among services.

The research shall tell how CPU-intensive apps (that are containerized on Kubernetes by the within the AWS infrastructure) behave in comparison to its performance [2]. To accomplish the best utilization of these types of application in the cloud, the article is going to shed a light on the scalability of Kubernetes under widely varying CPU limits and demands. By narrowing on a few specific performance consequences of CPU-intensive applications deployed on Kubernetes in the context of the AWS, the research will fill a gap in the scientific literature. However, Kubernetes, Docker containers and cloud platform possesses something in common as the past study, which are the lack of the full exploration of the impacts of CPU constraints and requests during the deployment of Kubernetes application. Through providing valuable advice on how to take full advantage of the Kubernetes container platform on AWS public cloud infrastructure for CPU-intensive applications, this research aims to reduce misconceptions.

Miika Moilanen's thesis focuses on deploying applications with Docker and Kubernetes, creating a Dockerfile for base image setup and Kubernetes deployment for testing[3]. Sponsored by Sparta Consulting Ltd., it aims to streamline deployment within the development team.The study addresses the practical use of Docker and Kubernetes for deployment, filling a gap in real-world application deployment research, enhancing testing efficiency and deployment processes.

The paper focuses on auto-scaling a defense application across the cloud using Docker and Kubernetes [4]. It highlights the integration of Docker for containerization and Kubernetes for orchestration to achieve scalability in cloud environments.This study addresses the lack of detailed research on auto-scaling defense applications with Docker and Kubernetes in cloud settings, providing insights into their combined use for efficient deployment and management.

The paper focuses on this particular DevOps approach for auto-deployment of web applications to multiple clouds is crafted using software tools like Ansible, Terraform, and others such as Kubernetes, Docker, and Cloud Servers which contribute greatly towards increasing productivity and reducing operational expenses[5]. It offers absolute uptime and security, thanks to the cloud hosting capabilities that deploy and keep a close eye on the web applications on the cloud. The abovementioned system serves the purpose of web app deployment but excludes the help for mobile and desktop applications which points to the fact that it is not a very versatile system.

The paper focuses on the ILP optimization formulations and the network-aware heuristics to budget the pod scheduling for the container-based applications while they are spread out in heterogeneous virtual machines geographically [6]. The main property of this method is reducing the time of adaptation and VM cost, at the same time managing the placement of the running containers. This study focuses on the process of scaling activities to a time and space scale in containerized application deployment, which is less studied.

The paper focuses on a framework for speed des construction of web applications and for optimizing response time that will improve the performance of web applications[7]. Containerization (using the Kubernetes service) and caching (using Redis cache) will be used; therefore the website will be more available and able to scale. Technologies that would assist in website performance like Kubernetes and Redis help in optimization and getting better codes of availability, scalability, and load balancing. Thus,

the study did this comparison to provide extrinsic motivation for people of low incomes to reach up to the Existing devices through shallow investment costs. The journal shows that it is necessary to be in possession of such a system that is efficient and ensuring quick response time in the performance of website functions and therefore interprets the significance of the technologies like Kubernetes and redis cache in website delivery of fast responsiveness. The framework that is proposed seeks to improve the performance of web services through both containerization and caching, while connection security with website speed and physical structure are made cheaper in fulfilling the demand of solutions.

The paper focuses on the process of placing container applications over cloud published VMs using ILP formulations and network aware heuristics within Kubernetes[8]. It targets fine-tune pod scheduling and cut adaptation time as well as VM expenses for effective rollout. Suggests the necessity of the ILP formulation and heuristics optimisation in order to improve both the scheduling et pod realization on Kubernetes.

The paper is focused on the improvement of the performance of web services by making a framework which exploits containerization and Kubernetes and caching with Redis to decrease reaction time and increase web site availability and scalability [9]. The research poster is based on a comprehensive evaluation of the framework's structure and testing and it proves the reduction of response times and expenses concurrently compared to the existing solutions, therefore, it provides a good basis for the improvement of web service efficiency from different points of view. The paper emphasizes the need for an integrated environment involving containerization and caching layers to enhance the web site performance, namely by reducing the overall response time. The research stresses the incorporation of advanced technologies such as Kubernetes and Redis cache to have a role in reducing costs in implementing response time boosting web service solutions.

This paper maps out the timeline of cloud computing and focus on how containerization has been useful in solving problems that developers face as a result of different environments [10]. About how using containerization such as Docker and Kubernetes in the application deployment, how containers work and scale and do load balancing the concept of how applications and their all dependencies can go into and run on lightweight isolated containers sharing one OS on one VM or server. It is necessary to underline the main benefits that can be received using microservices architecture, the importance of Kubernetes in the automation of containers orchestrations, features of Rancher as an element of multi-cluster orchestrations. It also offers a much clearer example of how to install Kubernetes cluster with Docker runtime environment, then specifically narrating the steps to install Kubernetes.

The paper is aimed at outlining the difficulties of compliance enforcedly inherent in Cloud-Native Applications (CNAs) [11]. It presents a reference architecture for observability and compliance pipeline to CNAs in integrated single/multi-cloud environments. This architecture embeds observability and compliance into CNAs which operate in small and large-scale settings in the mostly non- regulated plus some regulated industries thus enabling pioneers to develop products without worrying much about the compliance tasks.

ADF helicopter pilot training process continues the line of trainee’s entry, learning progression and instructors which pose a challenge due to the complexity in their dynamics. ATHENA, a strategic discrete event simulation tool, has been built to help in this aspect in workforce planning for this continuum[12]. The paper presents ATHENA and outlines how, by harnessing container technologies particularly Kubernetes acquiring Cloud infrastructure, it auto-scales to optimize the portioning and training of the effects.

The paper focuses on the discussion of Kubernetes as a critical open-source container orchestration tool and intends to provide a comprehensible bibliometric analysis of the topic[13]. It draws attention to Kubernetes as one of the most popular subjects in today’s research sphere while also considering cloud/fog/edge computing, IoT, containers, Docker, resource scheduling, microservices, and AI. Based on the

findings of this study, it is recommended that automation, 5G, scalability, serverless, service mesh, as well as blockchain should be considered as focus areas for future research in the field of Kubernetes to help academics and practitioners recognize the new tendencies in the research field.

III. METHODOLGY

To deploy the application, you use a manifest file to create all the objects required to run the AKS Store application. A Kubernetes manifest file defines a cluster's desired state, such as which container images to run. The manifest includes the following Kubernetes deployments and services:

Store front: Web application for customers to view products and place orders.
Product service: Shows product information.
Order service: Places orders.
Rabbit MQ: Message queue for an order queue.

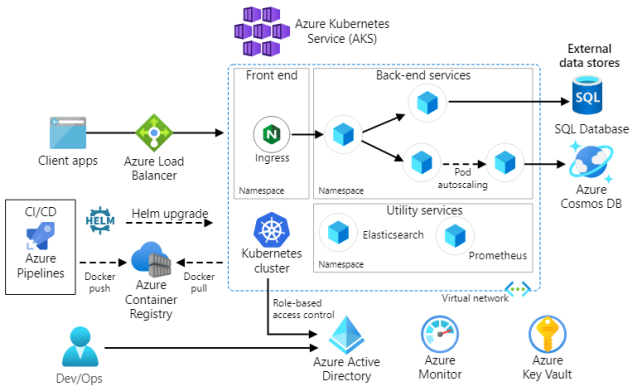

Figure 1: System Architecture Of AKS

Figure 1 illustrates a typical Azure Kubernetes Service (AKS) architecture, integrating various Azure services and tools. Client applications communicate with backend services via an Azure Load Balancer and an ingress controller within the AKS cluster. The cluster, organized into namespaces, hosts front-end, back-end, and utility services, with autoscaling capabilities. It interacts with external data stores like SQL Database and Azure Cosmos DB. Role-based access control is managed by Azure Active Directory, while Azure Monitor and Azure Key Vault handle monitoring and secret management, respectively. DevOps teams oversee this entire process, ensuring efficient deployment and management of containerized applications.

Preparation and Setup:

- Before continuing, make sure that you have a set of Kubernetes basics in mind and also Azure subscription.
- As a first step, make yourself knowledgeable about the Azure Cloud Shell environment and confirm whether the used identity is assigned adequate permissions for managing AKS.
- Before deploying a production-ready cluster, it is crucial to revise the baseline reference architecture that was discovered during the previous stage.

Cluster Deployment:

- Switch to the Azure portal and go to the creation of an AKS cluster.
- To make it easy and get familiar with the solution, deploy the cluster using default configuration parameters in order to evaluate and test the possibilities.
- Discuss the advantages associated with the Azure Linux node pool and how to integrate it.

Application Deployment:

- Generate an updated Kubernetes manifest file in which you describe how you want your app to look in the cluster.
- Replace the placeholders for the image names with the actual names of images hosted in your Azure Container Registry in manifest.yml.
- Signed the application and check the status of the deployment by using the kubectl apply command.

Testing and Verification:

- Check pods using Kubectl with the command- kubectl get pods to ensure that the resources have been created successfully in the AKS cluster.

- Use the command that will help to control the exposure of the application front end to the internet which is: kubectl get service.

- Make certain that the application will perform effectively and will be used by the users.

IV. RESULT & ANALYSIS

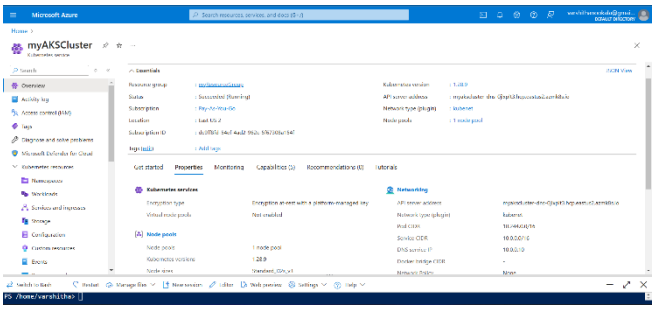

Figure 2: Details of the cluster created.

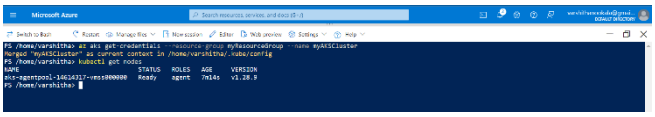

Figure 3: Connection of the clusters using nodes

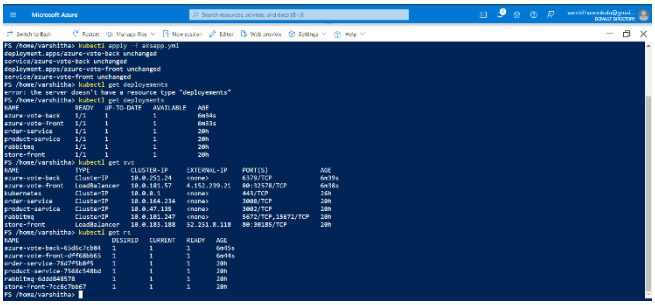

Figure 4: Creation of .yml in the directory

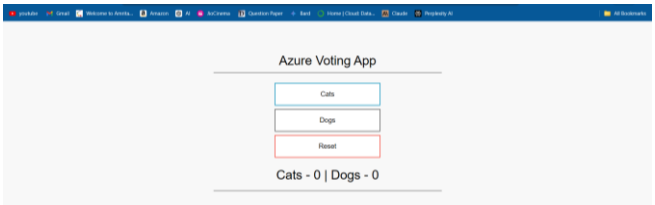

Figure 5: Deployed website using Kubernetes

Figure 2 illustrates the details of the cluster we created. The image shows the important details of the cluster like node pool details, networking properties of the cluster.

Figure 3 illustrates how we are connecting the clusters by Configuring kubectl to connect to your Kubernetes cluster using the az aks get-credentials command. This command downloads credentials and configures the Kubernetes CLI to use them.
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster.
Verify the connection to your cluster using kubectl get to return a list of the cluster nodes.
kubectl get nodes
After using these commands we get the output of the nodes which means the clusters are connected.

Figure 4 illustrates how to write the yaml code in the directory, for that we use ls then create a file in the directory vi aksapp.yml, then we use kubectl apply command and specify the yml file we created. Now to test the application we use kubectl get pods command, then we use kubectl get deployments to deploy our webapp. Then we get svc details then copy the external IP and paste it in the browser, which leads to the final website.

Figure 5 illustrates the output of the website we deployed using Kubernetes.

V. CONCLUSION

In conclusion, deploying an Azure Kubernetes Service (AKS) cluster through the Azure portal offers a powerful and efficient solution for managing containerized applications. AKS simplifies the orchestration of containers, streamlines the deployment process, and provides scalability and reliability for your applications. By leveraging AKS, organizations can benefit from a robust, cloud-native platform that enhances agility, accelerates development cycles, and optimizes resource utilization. Embracing AKS empowers teams to focus on innovation and application development while Azure handles the complexities of container orchestration, making it a valuable tool for modern cloud-native environments.

REFERENCES

[1] Poniszewska-Marańda, A.; Czechowska, E. Kubernetes Cluster for Automating Software Production Environment. Sensors 2021, 21, 1910. https://doi.org/10.3390/s21051910

[2] MOGALLAPU, RAJA. "Scalability of Kubernetes Running Over AWS-A Performance Study while deploying CPU intensive application containers." (2019).

[3] Moilanen, Miika. "Deploying an application using Docker and Kubernetes." (2018).

[4] Altaf, Umer, et al. "Auto-scaling a defence application across the cloud using docker and kubernetes." 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion). IEEE, 2018.

[5] Chavan, Ms Amrapali, et al. "Automate the Deployment of Any Custom Web Applications Across Several Clouds using Kubernetes Clusters."

[6] Rossi, Fabiana, et al. "Geo-distributed efficient deployment of containers with Kubernetes." Computer Communications 159 (2020): 161-174.

[7] Goyal, S., and A. Gill. "DEPLOYING A FRAMEWORK TO IMPROVE THE PERFORMANCE OF A WEB SERVICE." Technology (IJEET) 12.5 (2021): 137-147.

[8] Banerjee, Kakoli, et al. "A Survey on Kubernetes Policy Report Custom Resource Definition Kube-Bench Adapter." Advances in Data and Information Sciences: Proceedings of ICDIS 2022. Singapore: Springer Nature Singapore, 2022. 315-322.

[9] Hardikar, Sanjay, Pradeep Ahirwar, and Sameer Rajan. "Containerization: cloud computing based inspiration Technology for Adoption through Docker and Kubernetes." 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC). IEEE, 2021.

[10] S. Hardikar, P. Ahirwar and S. Rajan, "Containerization: Cloud Computing based Inspiration Technology for Adoption through Docker and Kubernetes," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021, pp. 1996-2003, doi: 10.1109/ICESC51422.2021.9532917.

[11] Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T. and Miranskyy, A., 2023. A Reference Architecture for Observability and Compliance of Cloud Native Applications. arXiv preprint arXiv:2302.11617.

[12] S. Kho Lin et al., "Auto-Scaling a Defence Application across the Cloud Using Docker and Kubernetes," 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Zurich, Switzerland, 2018, pp. 327-334, doi: 10.1109/UCC-Companion.2018.00076.

[13] Carrión, C., 2022. Kubernetes as a standard container orchestrator-a bibliometric analysis. Journal of Grid Computing, 20(4), p.42.

[14] Deng, S., Zhao, H., Huang, B., Zhang, C., Chen, F., Deng, Y., Yin, J., Dustdar, S. and Zomaya, A.Y., 2024. Cloud-Native Computing: A Survey from the Perspective of Services. Proceedings of the IEEE.

[15] Dewi, L.P., Noertjahyana, A., Palit, H.N. and Yedutun, K., 2019, December. Server scalability using kubernetes. In 2019 4th technology innovation management and engineering science international conference (TIMES-iCON) (pp. 1-4). IEEE.