Containerization of a polyglot microservice application using Docker and Kubernetes

Vamsi Krishna Yepuri, Venkata Kalyan Polamarasetty, Shivani Donthi, Ajay Kumar Reddy Gondi

Kent State University

Abstract—**This project investigates the benefits of containerization technology in modern software development and deployment. The study emphasizes the advantages of using Kubernetes and Docker in the development process, including the easy packaging and deployment of microservices, efficient resource utilization, faster startup times, and greater scalability and flexibility. The project concludes by proposing a study that involves creating a polyglot microservice application using Java, Python, and JavaScript, containerizing it with Docker, and deploying it in Kubernetes. The study aims to evaluate service discovery and auto-scaling in distributed mode and compare the performance metrics with virtual machine and container. The results of this study can inform software development teams about the benefits of containerization in modern software development and deployment.**

Index Terms—**Docker, Kubernetes, Containers, Microservices, Polyglot**

I. INTRODUCTION

Containerization has transformed software development and deployment with Docker being one of the most widely adopted open-source platforms for building, shipping, and running distributed applications. However, as containerized applications become more complex, managing them at scale can be a challenge. Kubernetes, an open-source container orchestration system, provides a platform for managing and deploying containerized applications at scale. Its popularity is attributed to its ability to automate deployment, scaling, and management of containerized applications, with features like automatic load balancing, scaling, and self-healing. Trends in Kubernetes technology include the adoption of serverless computing, which enables developers to run their code in lightweight containers without managing infrastructure or container orchestration. Kubernetes also finds use in hybrid and edge computing, where computing resources are distributed across multiple devices and locations, demonstrating its versatility and adaptability to modern software development and deployment needs.

Public cloud providers have played a significant role in the adoption and advancement of container technology. They offer container services that allow developers to deploy, manage, and scale containerized applications, which have reduced the barrier of entry to container adoption, making them available to businesses of all sizes. These providers have also been instrumental in promoting container standards and enabling interoperability, contributing to the adoption of Kubernetes as the de facto standard for container orchestration. Standardization of container orchestration simplifies the deployment and management of containerized applications across different cloud providers and on-premises environments.

Containers provide a variety of benefits in microservices architecture, with one of the most significant being the ease of packaging and deploying microservices. Each microservice can be packaged as a container image, which abstracts it from the underlying infrastructure and makes it simpler to deploy and run the services across various environments without compatibility issues. For example, consider an e-commerce application based on microservices, with various services such as product catalog, shopping cart, payment processing, and order management. Each of these services can be containerized, making it easy to deploy and update them independently. This enables developers to add new features or modify existing ones without impacting other services.

Additionally, containers provide a high degree of isolation between microservices, minimizing conflicts and dependencies between them. By packaging each service as a container, they can each run their own instance of the required database, without impacting the other services in the system. As a result, containers provide greater agility, scalability, and consistency in a microservices project, allowing developers to focus on developing and updating microservices rather than the infrastructure and deployment processes.

The use of virtual machines (VMs) can present challenges when it comes to running applications, as VMs require their own operating system and consume significant amounts of CPU, memory, and storage resources, which can result in slower application performance and increased complexity. This can be solved through the use of container technology, which allows multiple containers to share the same operating system kernel, leading to better resource utilization and faster startup times for applications, and ultimately greater scalability, flexibility, and cost efficiency.

For instance, consider a data analytics application that needs to process a large amount of data quickly and efficiently. If the application runs on a virtual machine, it may suffer from slower performance due to the overhead and complexity of running a full operating system. By using container technology, however, developers can run multiple lightweight containers on the same host operating system, resulting in faster performance, lower resource usage, and greater scalability for the application, allowing for faster data processing and analysis.

Achieving container capabilities with virtual machines is challenging due to the fundamental differences between the two technologies. Containers are lightweight and efficient, sharing the same operating system kernel as the host machine, while virtual machines emulate an entire hardware environment, including a separate operating system, which can lead to higher resource usage and greater complexity. Additionally, virtual machines lack portability and flexibility as each one requires a specific configuration of hardware and operating system, making it difficult to move or replicate them across different environments. Containers, on the other hand, can be easily packaged and deployed on any host system that supports the same container runtime. Although virtual

machines have their own strengths and use cases, containers are preferred for most modern applications due to their greater efficiency, flexibility, and portability, and because they can start up much more quickly, making them better suited for modern application deployment and scaling.

The traditional monolithic application approach involves building a single, comprehensive app, which can be time-consuming to develop and maintain, making it challenging to achieve agility. To address these issues and achieve greater flexibility and ease of maintenance, the world is moving towards microservices. In the era of microservices, numerous independent modules are set up in traditional virtual machines, which can lead to conflicting libraries and make the process of provisioning, scaling, service discovery, load balancing, and deployment manual and time-consuming. To solve these problems, microservices can be packed in containers instead of virtual machines, which will help to address conflicts and create independence between each microservice. Workloads can be run in a scalable and distributed manner using Kubernetes, which can solve auto-scaling and service discovery issues. This project involves building a polyglot microservice application from the ground up using Java, Python, and JavaScript, containerizing it with Docker, and deploying it in Kubernetes to examine service discovery and auto-scaling in distributed mode, while comparing the performance differences between the containerized technique and the traditional virtual machine strategy.

In this project, we aim to compare the performance of virtual machines and containers using several metrics. Firstly, we will examine the resource utilization of the microservice application between the two technologies, analyzing the CPU, memory, and storage usage to determine which technology is more efficient. Secondly, we will compare the startup time and scaling capabilities of virtual machines and containers to see which is faster and more scalable. Thirdly, we will examine the storage space utilized by both virtual machines and containers to see which technology uses storage more effectively. Finally, we will evaluate the portability of both virtual machines and containers and their ability to be moved across different environments. By comparing these performance metrics, we aim to determine which technology is more suitable for our microservice application and gain insights into the strengths and weaknesses of each technology.