iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82286

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Deploying Microservices for an E-Commerce Website using Azure DevOps Pipeline

Dr. Phanindra Reddy K[1], Gundapu Somya Lakshmi[2], Goutami Chidambar Deshpande[3], K Sanjana[4], Anusha B[5]
[1]*Associate Professor at Ballari Institute of Technology and Management, Ballari*
[2, 3, 4, 5]*Student at Ballari Institute of Technology and Management, Ballari, India*

Abstract: ***Modern e-commerce applications require high scalability, availability, security, and rapid deployment capabilities to handle dynamic user traffic and continuous feature updates. Traditional monolithic deployment models often fail to satisfy these requirements due to tightly coupled components, limited scalability, and manual deployment procedures, resulting in downtime and operational inefficiencies. To overcome these challenges, this project presents a cloud-native microservices-based deployment framework for an e-commerce application using an automated Azure DevOps CI/CD pipeline. The primary objective of the proposed system is to design and implement a secure, scalable, and automated cloud infrastructure that supports continuous integration and continuous deployment with minimal manual intervention. The application is developed using a microservices architecture and deployed on Azure Kubernetes Service (AKS) within a private cluster environment. Infrastructure provisioning is automated using Azure Resource Manager (ARM) templates and parameter files to ensure consistency, reliability, and repeatability across deployment environments. Azure DevOps is integrated with Git repositories to automatically trigger build and deployment pipelines whenever source code changes are committed. The proposed system utilizes several Microsoft Azure services, including Azure Container Registry for container image management, Azure Application Gateway with Web Application Firewall for secure traffic routing, Azure Key Vault for secret management, Azure Redis Cache for performance optimization, Azure Service Bus for inter-service communication, and Azure Cosmos DB for scalable data management. The implementation successfully achieves automated deployments, improved scalability, enhanced security, reduced deployment time, and efficient traffic management. Any modification in the application source code is automatically built and deployed, with updates reflected through the Application Gateway. This project demonstrates an industry-oriented DevOps approach for deploying robust and scalable e-commerce applications and highlights the effectiveness of cloud-native microservices architecture integrated with Azure DevOps automation.***

Keywords: Microservices, Azure DevOps, CI/CD Pipeline, Azure Kubernetes Service, Cloud Computing, E-Commerce.

I. INTRODUCTION

The rapid growth of e-commerce platforms has significantly increased the demand for scalable, reliable, and highly available web applications. Modern e-commerce systems are expected to handle fluctuating user traffic, secure online transactions, continuous feature updates, and minimal downtime. Traditional monolithic architectures often struggle to meet these requirements due to tightly coupled components, limited scalability, and complex deployment processes. As application complexity increases, maintaining and updating monolithic systems becomes time-consuming and operationally challenging.

Microservices architecture has emerged as an effective solution for overcoming these limitations by decomposing applications into independently deployable and manageable services. Combined with cloud computing, containerization, and DevOps practices, microservices enable rapid development, fault isolation, scalability, and continuous delivery. Microsoft Azure provides a comprehensive cloud ecosystem that supports microservices deployment using services such as Azure Kubernetes Service (AKS), Azure DevOps, Azure Container Registry, and Application Gateway. These cloud-native services enable organizations to automate deployments, improve resource utilization, and enhance application reliability.

In conventional e-commerce deployment systems, infrastructure provisioning and application deployment are often performed manually or through semi-automated approaches. Such practices lead to inconsistencies across environments, increased deployment failures, downtime during updates, and security vulnerabilities. Furthermore, monolithic applications require redeploying the entire system even for small code changes, making continuous integration and frequent software delivery difficult to achieve. These challenges highlight the need for a secure, scalable, and fully automated deployment framework capable of supporting modern cloud-native applications.

©IJRASET: All Rights are Reserved

This project, titled "Deploying Microservices for an E-Commerce Website using Azure DevOps Pipeline," focuses on designing and implementing a cloud-native deployment architecture for a microservices-based e-commerce application. The proposed system automates infrastructure provisioning using Azure Resource Manager (ARM) templates and implements Continuous Integration and Continuous Deployment (CI/CD) pipelines through Azure DevOps integrated with Git repositories. Containerized microservices are deployed securely on Azure Kubernetes Service, while Application Gateway with Web Application Firewall ensures secure traffic management. Additional Azure services such as Azure Key Vault, Azure Redis Cache, Azure Service Bus, and Azure Cosmos DB are integrated to improve security, caching, messaging, and database management.

The primary motivation behind this project is the increasing industry adoption of DevOps methodologies and cloud-native microservices architectures. Organizations require deployment systems that can automatically build, test, and deploy applications with minimal manual intervention while ensuring high availability and operational efficiency. Implementing Azure DevOps pipelines, Kubernetes orchestration, and Infrastructure as Code practices provides practical exposure to real-world cloud technologies and deployment strategies. This work aims to bridge the gap between academic learning and industry-standard DevOps practices by developing a scalable and automated deployment solution for modern e-commerce applications.

The main objectives of this project include designing a microservices-based e-commerce architecture, automating infrastructure provisioning using ARM templates, implementing CI/CD pipelines using Azure DevOps, deploying secure containerized services on AKS, and improving deployment efficiency, scalability, and reliability. The project scope primarily focuses on cloud infrastructure automation, Kubernetes deployment, containerization, and secure DevOps workflows. Advanced business analytics, payment gateway implementation, and large-scale production traffic testing are beyond the scope of this work.

II. RELATED WORK

Mandi Walls in paper [1] presented Building a DevOps Culture, which focuses on improving collaboration between development and operations teams through organizational and cultural transformation. The study emphasizes setting achievable DevOps goals, assigning technical leadership, encouraging stakeholder communication, and implementing continuous feedback mechanisms. The proposed methodology includes cultural assessment, pilot project implementation, team training, and workflow optimization. The implementation demonstrated improved collaboration, streamlined deployment processes, faster release cycles, and increased operational efficiency. However, the study lacks quantitative evaluation metrics and provides limited guidance regarding specific DevOps tools and implementation strategies. Future work suggested includes empirical analysis of DevOps adoption, development of scalable frameworks, and integration of automated tooling for large-scale organizations.

Sheldon Smith et al. in paper [2], titled Benchmarks for End-to-End Microservices Testing, addressed the challenges associated with testing distributed microservices-based systems. The authors proposed a benchmark suite for end-to-end testing using open-source applications such as TrainTicket and eShopOnContainers. Functional and load-testing scenarios were implemented using tools such as Selenium along with service dependency graphs and tracing techniques. The results demonstrated effective validation of performance, reliability, and system bottlenecks in microservices environments. However, the research mainly focused on functional and load testing while giving less importance to security and fault tolerance aspects. The dependency on specific testing tools also reduced flexibility. Future enhancements proposed by the authors include integrating resilience testing, security evaluation, and automated test generation within CI/CD environments for continuous quality assurance.

Sergio Moreschini et al. in paper [3] explored the application of Artificial Intelligence techniques across the microservices life cycle with special emphasis on DevOps practices. The study systematically analyzed how AI contributes to improving quality attributes such as performance, reliability, and operational efficiency in microservices systems. Using systematic mapping and literature review methodologies, the authors examined 269 research studies and identified multiple AI-driven approaches applied during different DevOps phases. The findings revealed that AI is primarily used in operational monitoring, clustering, automation, and system optimization. However, limited focus was observed in areas such as usability, portability, and explainability of AI models. The study also highlighted the lack of industry-based practical implementations. Future work includes extending research towards AIOps and MLOps, integrating generative AI tools for DevOps automation, and improving the interpretability of AI-based systems for broader adoption in enterprise environments.

The Microsoft Azure Architecture Center in paper [4], titled Microservices Assessment and Readiness, presented a structured framework for organizations planning to migrate from monolithic systems to microservices architectures. The guide focuses on evaluating business priorities, team structures, architectural decisions, DevOps readiness, and infrastructure scalability before adopting microservices.

It recommends decomposition strategies such as the Strangler Fig pattern and emphasizes the importance of continuous integration, continuous deployment, and Infrastructure as Code practices. The framework also highlights the need for cross-functional collaboration and governance during migration. Although the guide provides a comprehensive approach, it identifies challenges related to data synchronization, organizational restructuring, and migration complexity. Future improvements suggested include developing detailed readiness metrics, migration assessment tools, and real-world case studies for large-scale enterprise transformations.

III. METHODOLOGY

The proposed system follows the Agile development model with iterative implementation and testing. The system architecture is based on microservices, where each service is containerized using Docker and deployed on Azure Kubernetes Service. The workflow begins when developers commit source code changes to the Git repository integrated with Azure DevOps. Azure DevOps pipelines automatically trigger build and deployment stages. Docker images are created and stored in Azure Container Registry. Kubernetes then deploys the updated microservices to AKS clusters. Azure Application Gateway with Web Application Firewall routes incoming traffic securely to backend services. Azure Key Vault manages secrets and credentials securely, while Azure Redis Cache improves application performance. Azure Monitor and Log Analytics continuously monitor system health and performance.

A. System Architecture

An Architecture Diagram provides a high-level representation of the overall structure of a software system by illustrating its major components, subsystems, and interactions. It helps developers and stakeholders understand how different modules communicate, how data flows through the system, and how the application infrastructure is organized before implementation. The architecture diagram of the proposed system presents a complete view of the cloud infrastructure and DevOps workflow implemented for the microservices-based e-commerce application.

In the proposed architecture, developers push application source code to the Azure DevOps Git Repository, which acts as the centralized version control system. Whenever code changes are committed, the Azure DevOps CI/CD pipeline is automatically triggered to build Docker container images for the microservices. These images are securely stored in Azure Container Registry (ACR) with proper version management. The containerized microservices are then deployed into the Azure Kubernetes Service (AKS) cluster, which manages scalability, load balancing, fault tolerance, and high availability of the application services.

Incoming user requests from the internet are routed through Azure Application Gateway integrated with Web Application Firewall (WAF), which filters malicious traffic and securely forwards valid requests to the appropriate microservices deployed in AKS. For enhanced security, Azure Firewall controls inbound and outbound traffic rules, while sensitive information such as secrets, certificates, and access keys are securely managed using Azure Key Vault. The backend services interact with Azure Cosmos DB for database operations, Azure Redis Cache for performance optimization, and Azure Service Bus for reliable asynchronous communication between microservices. Additionally, Azure Monitor and Log Analytics continuously collect system logs, metrics, and performance data to support monitoring, troubleshooting, and optimization. Overall, the architecture diagram demonstrates a secure, scalable, and automated cloud-native deployment framework using Microsoft Azure services.

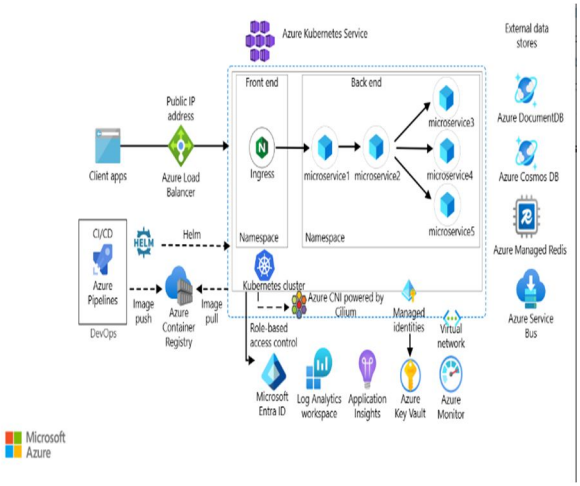


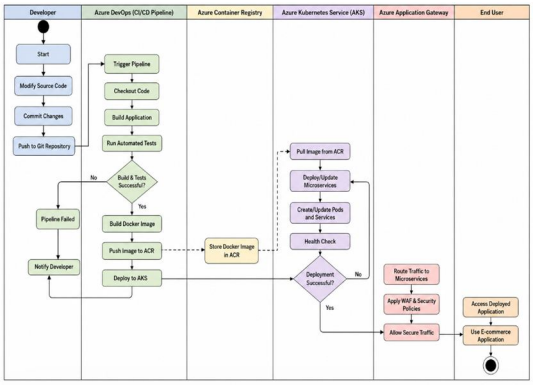
Fig. 1. Azure DevOps-Based Microservices Architecture for E-Commerce Application

B. Development Model

The proposed system follows the Agile Software Development Model because of its flexibility and iterative development approach. The entire project is divided into multiple modules such as infrastructure provisioning, CI/CD automation, Kubernetes deployment, monitoring services, security configuration, and cloud resource management. Each module is implemented, tested, and refined independently before integration into the complete system. The Agile model supports continuous improvement, rapid deployment cycles, efficient issue handling, and better collaboration between development and operations processes. This development methodology also helps reduce deployment risks while improving scalability, automation efficiency, and system reliability.

C. System Workflow

The workflow of the proposed deployment system begins when developers modify application source code or deployment configuration files and push the updated code into the Git repository integrated with Azure DevOps. Once changes are committed, the Azure DevOps CI/CD pipeline is automatically triggered. The pipeline performs source code checkout, build operations, and automated creation of Docker container images for the microservices. These images are then pushed into Azure Container Registry (ACR) for secure version-controlled storage. Azure Kubernetes Service (AKS) retrieves the latest container images and deploys or updates the microservices within the Kubernetes cluster. Incoming traffic from end users is routed securely through Azure Application Gateway with Web Application Firewall protection. At the same time, monitoring tools collect logs, metrics, and deployment status information continuously for performance analysis and troubleshooting.



.Fig. 2. Workflow of Automated Azure DevOps Deployment System

D. CI/CD Pipeline and Deployment Automation

At the core of the proposed system lies the automated Continuous Integration and Continuous Deployment pipeline implemented using Azure DevOps. Whenever developers commit code changes, the pipeline automatically validates the source code, executes build operations, and creates Docker images for the updated services. Infrastructure provisioning is automated using Azure Resource Manager (ARM) templates, ensuring consistency across environments. Deployment automation reduces manual intervention and minimizes downtime during application updates. Azure Kubernetes Service orchestrates the deployment process and ensures scalability, fault tolerance, and high availability of the microservices. All deployment activities are continuously monitored using Azure Monitor and Log Analytics, enabling rapid detection and resolution of issues.

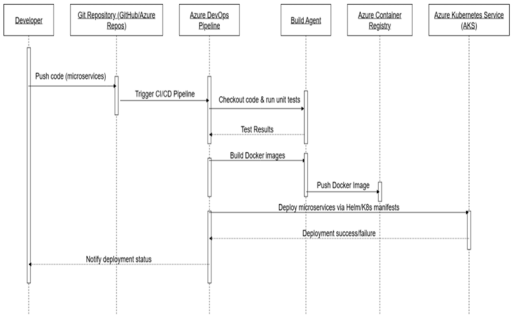


Fig. 3. Sequence Diagram for CI/CD Deployment Workflow

E. Security and Monitoring Module

The Security and Monitoring Module is designed to improve system reliability, application protection, and deployment observability. Azure Key Vault securely stores secrets, access keys, and certificates required for deployment and application communication. Azure Firewall and Web Application Firewall (WAF) protect the application from unauthorized access and malicious traffic. Azure Monitor and Log Analytics continuously collect logs, metrics, and health data from the deployment environment, helping administrators track application performance and troubleshoot issues efficiently. These monitoring and security mechanisms ensure that the deployed e-commerce application remains secure, scalable, and operationally stable.

F. Database Design

The proposed system utilizes Azure Cosmos DB for scalable cloud database management and Azure Redis Cache for high-performance caching. The database structure stores user information, product details, order records, deployment logs, monitoring metrics, and application configuration data securely. Relationships between services and resources are managed efficiently to support rapid data retrieval and scalable application performance. Azure Service Bus enables reliable asynchronous communication between distributed microservices, ensuring smooth interaction between independent services within the deployment architecture.

G. Tools and Technologies

The implementation of the proposed system utilizes several modern cloud and DevOps technologies. Microsoft Azure serves as the primary cloud platform for deployment and infrastructure management. Azure DevOps is used for implementing Continuous Integration and Continuous Deployment pipelines, while Git and GitHub provide version control support. Docker is used for containerization, and Azure Kubernetes Service (AKS) manages orchestration and scaling of containerized microservices. Azure Container Registry stores Docker images securely. Security services such as Azure Key Vault, Azure Firewall, and Application Gateway with Web Application Firewall provide secure communication and traffic management. Azure Cosmos DB, Azure Redis Cache, Azure Service Bus, Azure Monitor, and Log Analytics further enhance database management, caching, messaging, monitoring, and deployment reliability within the proposed cloud-native system.

IV. RESULTS AND DISCUSSION

The proposed system was successfully implemented and deployed using Microsoft Azure cloud services and Azure DevOps automation pipelines. The results obtained from the implementation demonstrate the effectiveness of the microservices-based deployment architecture in achieving automated application deployment, scalability, security, and operational reliability. The deployed system successfully integrates Azure DevOps CI/CD pipelines, Azure Kubernetes Service (AKS), Azure Container Registry (ACR), Application Gateway, and multiple Azure cloud services to provide a complete cloud-native deployment environment for the e-commerce application.

A. Azure DevOps Pipeline Execution

The Azure DevOps Pipeline Execution screen demonstrates the successful execution of the Continuous Integration and Continuous Deployment workflow triggered automatically whenever source code changes are committed to the Git repository. The pipeline performs multiple automated stages including source code checkout, application build, Docker container image creation, testing, and deployment to Azure Kubernetes Service. The successful execution of these stages confirms that the CI/CD automation framework functions correctly and enables reliable software delivery with minimal manual intervention.

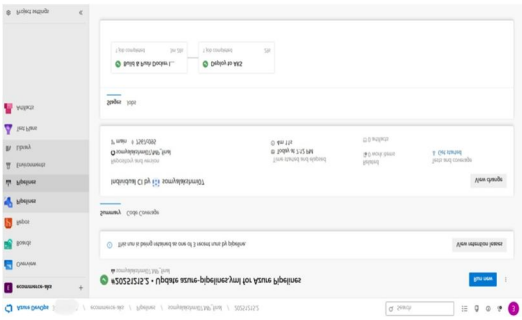


Fig. 4. Azure DevOps Pipeline Execution Screen

B. Azure Container Registry

The Azure Container Registry screen displays the Docker container images generated by the CI/CD pipeline and pushed securely into the registry. The successful storage and versioning of container images confirm that the containerization process works effectively within the deployment framework. Azure Container Registry acts as a centralized repository for managing container images used during Kubernetes deployment operations.

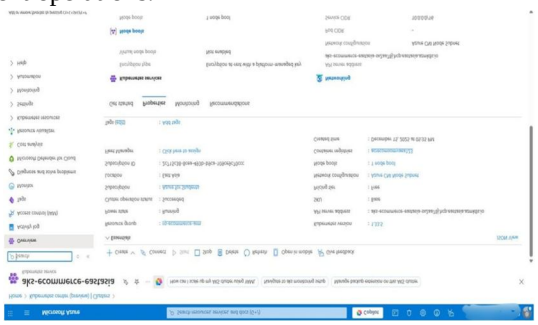


Fig. 5. Azure Container Registry Screen

C. Kubernetes Deployment Status

The Kubernetes Deployment Status screen shows the running pods, deployments, and services within the Azure Kubernetes Service cluster. The results confirm that the microservices are deployed successfully and remain active within the Kubernetes environment. Azure Kubernetes Service provides scalability, fault tolerance, automatic orchestration, and load balancing capabilities, ensuring stable application performance and high availability.

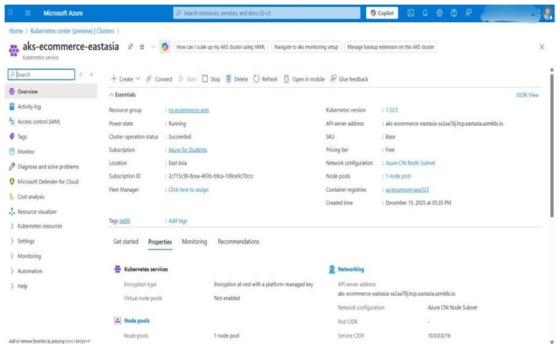


Fig. 6. Kubernetes Deployment Status Screen

D. Application Gateway Configuration

The Application Gateway Configuration screen presents the configured Azure Application Gateway integrated with Web Application Firewall (WAF) rules for secure traffic management. The Application Gateway successfully routes incoming user requests to the appropriate microservices running inside the AKS cluster while filtering malicious traffic and unauthorized access attempts. This configuration improves network security and ensures secure communication between users and application services.

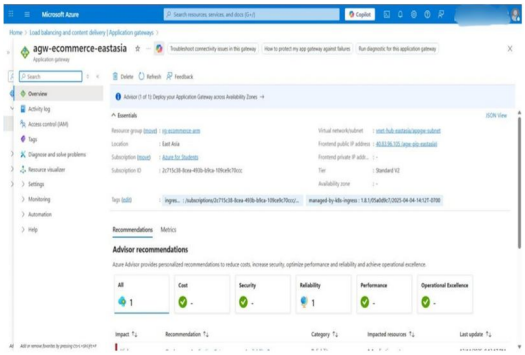


Fig. 7. Application Gateway Configuration Screen

E. Deployed Application Output

The Deployed Application Output screen displays the live e-commerce application accessed through the Application Gateway endpoint after successful deployment. The results confirm that the CI/CD pipeline, Kubernetes deployment, and secure traffic routing mechanisms function correctly. End users can access the deployed application with minimal downtime, demonstrating the effectiveness of the automated deployment architecture.

F. Performance Analysis

The performance of the proposed system was analyzed and compared with traditional monolithic and manual deployment approaches. The proposed automated deployment framework significantly reduces deployment time and operational complexity compared to existing deployment methods. Automated CI/CD pipelines eliminate manual deployment tasks, reducing human errors and ensuring deployment consistency across environments. Kubernetes-based deployment improves scalability through automatic scaling capabilities, while Application Gateway enhances application availability and secure traffic handling.

The proposed system also minimizes downtime during application updates because updated container images are automatically deployed into the Kubernetes cluster without requiring complete system shutdowns. Infrastructure automation using Azure Resource Manager (ARM) templates ensures consistent resource provisioning and reduces configuration mismatches across deployment environments. Compared to traditional deployment systems, the proposed architecture demonstrates better scalability, reliability, maintainability, and deployment efficiency.

G. Discussion

The results obtained from the implementation clearly demonstrate that the proposed microservices deployment architecture successfully satisfies both functional and non-functional requirements of the e-commerce application. The integration of Azure DevOps CI/CD pipelines automates the complete deployment lifecycle, enabling faster and more reliable application delivery. Infrastructure automation using ARM templates reduces manual configuration efforts and ensures consistent cloud resource provisioning.

Security within the proposed system is enhanced through Web Application Firewall integration, private networking, Azure Firewall policies, and centralized secret management using Azure Key Vault. The deployment framework also improves scalability and operational efficiency through Azure Kubernetes Service auto-scaling and container orchestration features. Any modification made to the source code is automatically integrated, tested, containerized, and deployed with minimal manual intervention, thereby reducing deployment downtime and operational complexity.

Overall, the proposed system achieves its intended objectives effectively and demonstrates a practical industry-oriented DevOps solution for deploying scalable and secure cloud-native e-commerce applications. Future improvements may include advanced monitoring dashboards, AI-driven auto-healing mechanisms, predictive scaling strategies, and large-scale performance testing to further enhance deployment reliability and operational intelligence.

V. CONCLUSIONS AND FUTURE WORK

This project addressed the challenges associated with deploying and managing modern e-commerce applications using traditional monolithic and manual deployment approaches, which are often time-consuming, error-prone, difficult to scale, and operationally inefficient. To overcome these limitations, a cloud-native microservices deployment framework was designed and implemented using Microsoft Azure cloud services and Azure DevOps automation pipelines. The proposed system successfully established an automated deployment environment in which microservices are containerized using Docker, orchestrated through Azure Kubernetes Service (AKS), and deployed using a fully automated Continuous Integration and Continuous Deployment (CI/CD) pipeline.

Infrastructure provisioning was automated using Azure Resource Manager (ARM) templates, ensuring consistency, repeatability, and reliable resource configuration across deployment environments. Whenever modifications are made to the application source code, the Azure DevOps pipeline is automatically triggered to build, test, containerize, and deploy the updated microservices into the Kubernetes cluster. Application Gateway integrated with Web Application Firewall securely routes user traffic while protecting the deployed services from malicious requests and unauthorized access. During implementation and testing, the proposed system demonstrated reliable performance across multiple deployment stages, including build validation, deployment automation, Kubernetes orchestration, and application accessibility. The CI/CD pipeline executed successfully without manual intervention, and application updates were reflected with minimal or zero downtime. The deployment framework significantly improved scalability, deployment efficiency, security, fault tolerance, and maintainability compared to traditional deployment approaches.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Overall, the project successfully demonstrates an effective industry-oriented DevOps solution for deploying scalable and secure e-commerce applications using cloud-native microservices architecture. The integration of Azure cloud services, Kubernetes orchestration, automated CI/CD pipelines, and infrastructure automation provides a strong foundation for enterprise-level cloud deployments and modern software delivery practices.

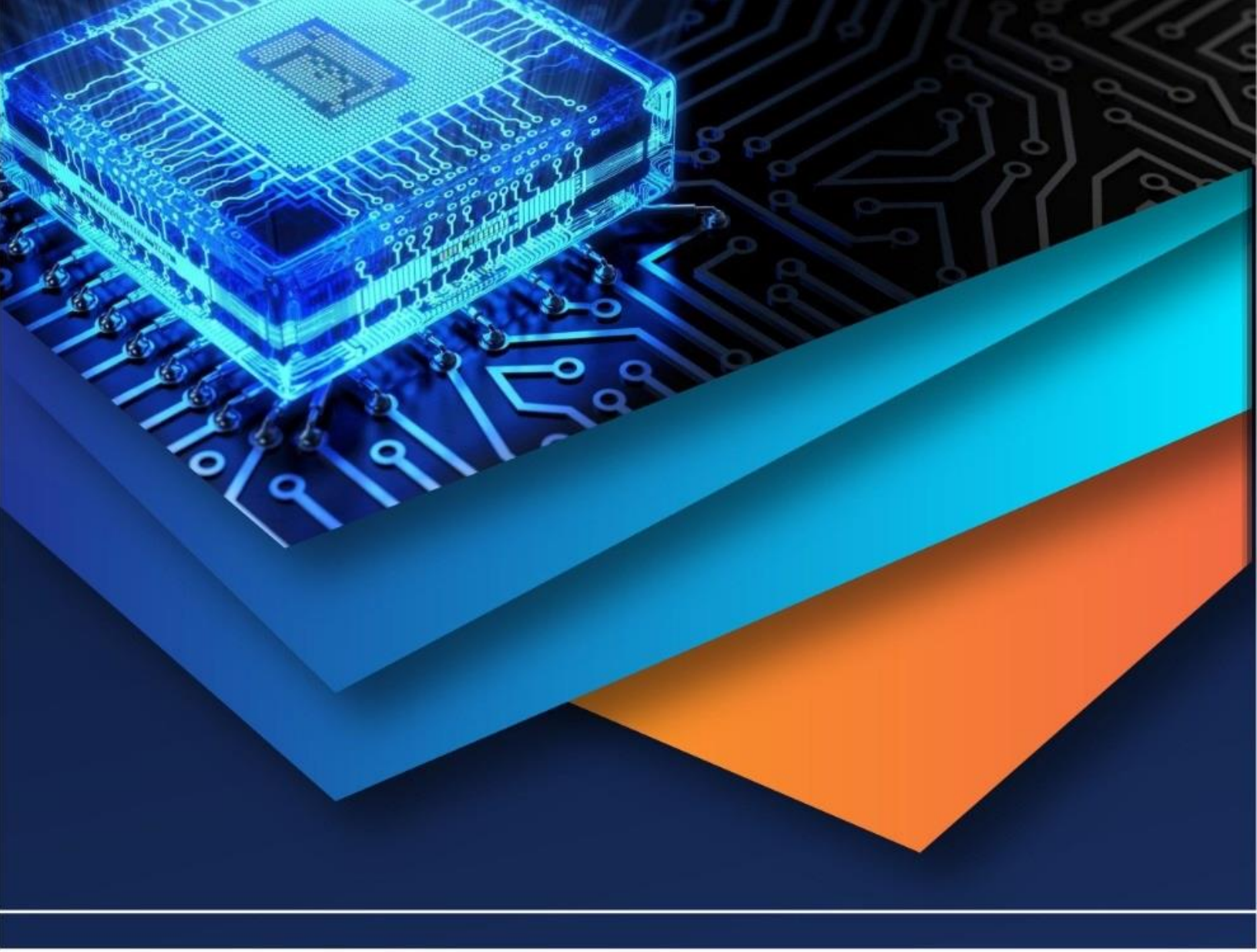
Although the proposed system successfully achieves automated deployment, scalability, and secure cloud-native application management, several enhancements can be incorporated in the future to further improve system capabilities and operational intelligence. Advanced monitoring and visualization tools such as Prometheus and Grafana can be integrated to provide detailed real-time insights into application performance, infrastructure health, and resource utilization. Auto-healing mechanisms and advanced Kubernetes autoscaling policies can also be implemented to improve system fault tolerance and service availability during high traffic conditions or unexpected failures.

The system can be further extended by integrating Artificial Intelligence and Machine Learning-based analytics for predictive resource management and traffic pattern analysis. AI-driven monitoring systems can help optimize cloud resource allocation, reduce infrastructure costs, and improve application performance dynamically based on user demand. Support for mobile application deployment and multi-region Kubernetes clusters can also be incorporated to improve global accessibility, low-latency communication, and disaster recovery capabilities.

Additional security enhancements such as Zero Trust Architecture, advanced threat detection systems, intelligent intrusion prevention mechanisms, and automated security compliance monitoring can further strengthen the deployment environment. Future research can also explore migration toward fully serverless microservices architectures or multi-cloud deployment strategies to improve flexibility, resilience, portability, and vendor independence. These enhancements would make the proposed deployment framework more suitable for large-scale enterprise-level cloud applications and highly distributed production environments.

REFERENCES

[1] M. Walls, Building a DevOps Culture, O'Reilly Media, 2018.
[2] S. Smith et al., "Benchmarks for End-to-End Microservices Testing," IEEE Publications, 2021.
[3] S. Moreschini et al., "AI Techniques in the Microservices Life Cycle," Journal of Systems and Software, 2022.
[4] Microsoft Azure Architecture Center, "Microservices Assessment and Readiness Guide," Microsoft Docs, 2023.
[5] B. Burns and D. Oppenheimer, Designing Distributed Systems, O'Reilly Media, 2018.
[6] Docker Documentation, https://docs.docker.com/
[7] Microsoft Azure DevOps Documentation, https://learn.microsoft.com/en-us/azure/devops/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)