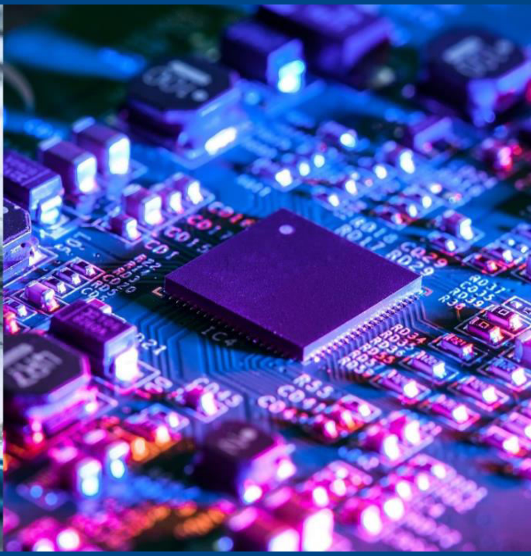
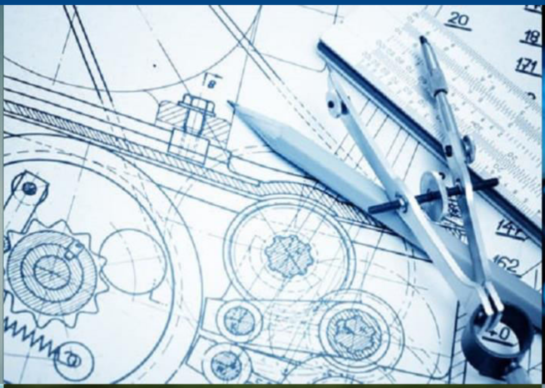
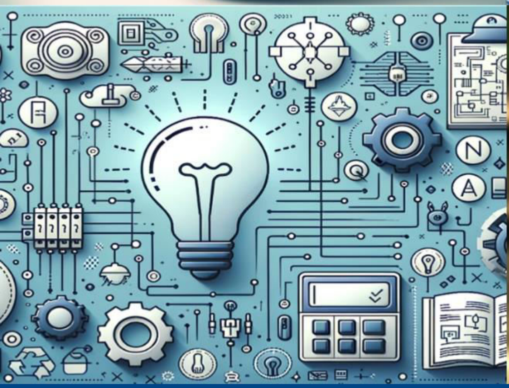
ISSN: 2582-7219



International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





Impact Factor: 8.206

Volume 8, Issue 6, June 2025

© 2025 IJMRSET | Volume 8, Issue 6, June 2025| DOI:10.15680/IJMRSET.2025.0806162

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Ecommerce Robot Store Deployment with DevOps

Aakash Balasaheb Shendage, Prof. Amruta Jawade

Post Graduate Student, Dept. of Master of Computer Applications, Anantrao Pawar College of Engineering and Research, Pune, India

Dept. of Master of Computer Applications, Anantrao Pawar College of Engineering and Research, Pune, India

ABSTRACT: In the rapidly evolving landscape of e-commerce, technological innovations are continuously reshaping the way we shop online. One such ground breaking advancement is the development of the **3-Tier Robot Shop Application**, a sophisticated system designed to enhance the overall shopping experience for users.

By integrating advanced automation, artificial intelligence, and machine learning, this application streamlines various aspects of online retail, from personalized recommendations and efficient order processing to seamless inventory management and secure transactions. The system not only improves user satisfaction through intuitive navigation and real-time assistance but also empowers retailers by providing valuable insights, optimizing logistics, and reducing operational costs.

At the core of the application lies the **three-tier architecture**—comprising the presentation tier, logic tier, and data tier—each responsible for a distinct layer of operation. The presentation tier offers a user-friendly interface, enabling customers to browse, search, and purchase products with ease. The logic tier handles business logic, integrating AI-driven engines that analyze user behavior to deliver tailored experiences. The data tier ensures robust data management, storing critical information such as user profiles, transaction histories, and inventory records in a secure and scalable environment**.**

I. INTRODUCTION

In today's fast-paced digital world, the e-commerce industry is undergoing a transformative shift driven by rapid advancements in technology. Consumers now expect faster, smarter, and more personalized shopping experiences, pushing retailers to adopt innovative solutions to stay competitive. One such innovation is the **3-Tier Robot Shop Application**—an intelligent, automated system that leverages **artificial intelligence (AI), machine learning (ML), and automation** to revolutionize the online shopping process.

This application is built on a robust three-tier architecture—comprising the presentation, logic, and data layers—that work together to deliver seamless user experiences, optimize backend operations, and ensure secure, real-time transactions. By integrating smart features such as personalized recommendations, automated inventory management, and real-time customer support, the 3-Tier Robot Shop Application sets a new standard for modern e-commerce platforms. E-commerce has become an essential part of modern retail, offering businesses a global reach and consumers the convenience of purchasing from anywhere. The robotics industry is simultaneously seeing rapid growth, leading to a need for specialized online platforms that cater to robotic products. However, deploying and maintaining such platforms using traditional software development methods often results in inflexible systems and long update cycles. The integration of DevOps into the development and deployment pipeline allows for a culture of collaboration between developers and operations teams. This paper discusses the complete deployment cycle of an e-commerce robot store and shows how DevOps tools like Docker, Kubernetes, Jenkins, Terraform, and Prometheus enhance productivity, reliability, and scalability.

II. LITERATURE REVIEW

DevOps is widely recognized as a transformative approach in modern software development. It bridges the gap between software development and IT operations, enabling continuous integration and continuous deployment

(CI/CD). Studies by Kim et al. (2016) suggest that organizations that adopt DevOps practices experience 46x more frequent deployments and 96x faster mean time to recovery. The microservices architecture, as discussed by Fowler (2012), allows systems to scale efficiently and isolate faults within individual services. The use of containerization (via Docker) and orchestration platforms like Kubernetes has further revolutionized application deployment in distributed environments. Combining these concepts into a practical application like an E-commerce Robot Store serves as a strong case for the advantages of DevOps.

The microservices architecture, as discussed by Fowler (2012), enables the development of loosely coupled services that can be deployed and scaled independently. This architecture supports fault isolation, meaning a failure in one microservice does not affect the entire system, which is critical for e-commerce platforms that demand high availability and resilience.

Containerization technologies like Docker have introduced a paradigm shift in application deployment by providing lightweight, portable, and consistent runtime environments across development, staging, and production. In conjunction, orchestration platforms such as Kubernetes manage container lifecycles, auto-scaling, load balancing, and self-healing of applications, ensuring robust and efficient operations at scale.

Moreover, Infrastructure as Code (IaC) tools such as Terraform and Ansible allow teams to automate and version-control the provisioning of infrastructure, reducing manual errors and improving consistency across environments. With cloud providers like AWS offering managed services (e.g., Amazon EKS, RDS, CloudWatch), the integration of DevOps into cloud-native applications has become more seamless, further enhancing agility and scalability.

III. THEORETICAL FRAMEWORK

The theoretical framework for the E-commerce Robot Store with DevOps integrates concepts from **software engineering, systems theory, cloud computing, and DevOps methodologies** to explain how modern development and operational practices can be applied to build scalable and efficient e-commerce platforms.

Systems Theory: The project is built on systems theory, which views the E-commerce Robot Store as an integrated system consisting of interconnected subsystems (frontend, backend, database, infrastructure, monitoring, etc.). Each subsystem performs a specific function but contributes to the overall performance and behavior of the platform. Systems theory helps in understanding dependencies, inputs, outputs, and feedback loops essential for continuous improvement and system reliability.

Agile Software Development: The development process is guided by Agile principles, which promote iterative development, continuous feedback, and customer collaboration. Agile methodology supports the flexible development of features and rapid adaptation to changing requirements, making it suitable for dynamic environments like e-commerce.

DevOps Culture and Practices: DevOps bridges the gap between software development (Dev) and IT operations (Ops), emphasizing collaboration, automation, integration, and continuous delivery. The DevOps approach is underpinned by the **CALMS model** (Culture, Automation, Lean, Measurement, Sharing), which serves as a foundation for implementing best practices across the software lifecycle:

- **Culture:** Encourages shared responsibility between developers and operations teams.
- **Automation:** Enables faster builds, tests, and deployments through tools like Jenkins and Terraform.
- **Lean:** Focuses on eliminating waste and optimizing workflows.
- **Measurement:** Uses metrics (e.g., uptime, MTTR) for performance improvement.
- **Sharing:** Promotes knowledge sharing and transparency within teams.

Microservices Architecture: The theoretical basis of using **microservices** lies in **Service-Oriented Architecture (SOA)**. Each feature or functionality in the store is built as an independent, deployable service. This modularity allows easier testing, scaling, and fault isolation. It also aligns with the DevOps goal of deploying smaller, incremental changes with minimal risk.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Containerization and Orchestration: Based on the principle of **virtualization and process isolation**, containerization using Docker provides a standard runtime environment, eliminating the "it works on my machine" problem. Kubernetes, based on control theory and distributed system management, manages the lifecycle of containers, scales workloads dynamically, and ensures high availability through self-healing capabilities.

IV. PROPOSED METHODOLOGY

The proposed methodology for developing and deploying the E-commerce Robot Store using DevOps follows a structured, iterative, and automation-driven approach. It integrates **Agile development**, **DevOps lifecycle**, and **cloud-native technologies** to ensure high availability, scalability, and faster release cycles. The methodology is divided into the following phases.

Requirement Analysis: In this phase, detailed functional and non-functional requirements are gathered by interacting with stakeholders. Functional requirements include user registration, product listing, cart management, payment integration, and order tracking. Non-functional requirements focus on security, performance, scalability, and system availability. Tools like Jira are used to manage user stories and track progress.

Development: The development phase adopts **Agile sprints** with CI/CD integration. Each microservice is developed using **Java Spring Boot** for backend and **Angular** for frontend. Code is version-controlled using GitHub. Unit testing and integration testing are performed at every stage. Code quality is ensured using tools like **SonarQube**.

Containerization: All microservices are containerized using **Docker**. Each container encapsulates its service, dependencies, and runtime environment. This ensures platform-independent deployment and simplifies dependency management.

Infrastructure Provisioning: Infrastructure is provisioned using **Infrastructure as Code (IaC)** tools:
- **Terraform** for setting up AWS services (e.g., EC2, RDS, S3, EKS)
- **Ansible** for server configuration and deployment automation

This ensures repeatable, consistent, and version-controlled infrastructure setup.

Continuous Integration and Continuous Deployment (CI/CD): Automated CI/CD pipelines are created using **Jenkins**:
- **Build** → Code is compiled and packaged.
- **Test** → Unit and integration tests are executed.
- **Scan** → Security and code quality scans are performed.
- **Push** → Docker images are pushed to DockerHub or Amazon ECR.
- **Deploy** → Kubernetes deploys the images to AWS EKS using Helm charts.

This reduces manual intervention and accelerates delivery.

V. COMPARATIVE ANALYSIS

The E-commerce Robot Store deployed using DevOps is compared against traditional software deployment models to highlight improvements in scalability, speed, reliability, and maintainability. The following table and explanation provide a clear comparison between **Traditional Development & Deployment** and the **DevOps-based Approach** used in this project.

Criteria	Traditional Deployment	DevOps-Based Deployment
Development Approach	Waterfall or siloed	Agile + Continuous Integration
Release Frequency	Monthly or Quarterly	Daily or Weekly (Automated CI/CD)
Infrastructure Provisioning	Manual, time-consuming	Automated using Infrastructure as Code (Terraform)
Scalability	Limited, requires manual scaling	Auto-scaling with Kubernetes
Downtime during	High, due to full-system restarts	Minimal, with rolling or blue-green deployments

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Criteria	Traditional Deployment	DevOps-Based Deployment
Deployment		
Monitoring	Basic server monitoring	End-to-end monitoring with Prometheus & Grafana
Collaboration	Developers and Ops work separately	Unified Dev + Ops teams with shared ownership
Testing	Manual testing, less coverage	Automated testing in pipeline
Fault Recovery	Manual, time-intensive	Self-healing systems and quick rollback
Security	Often added later	Integrated into DevSecOps pipeline
Cost Optimization	Over-provisioning common	Pay-as-you-go, optimized with cloud auto-scaling

Key Insights:

1. **Release Cycle Efficiency**: Traditional systems follow rigid release cycles, leading to infrequent updates. In contrast, the DevOps-enabled system supports continuous delivery with faster time to market through automated pipelines.
2. **Infrastructure Management**: Manual provisioning in traditional systems increases setup time and human errors. DevOps leverages Terraform and Ansible to automate and version infrastructure, enabling rapid, repeatable, and error-free deployments.
3. **Scalability and Reliability**: Using Kubernetes, the E-commerce Robot Store can automatically scale services during peak traffic (e.g., flash sales) and heal failed services without manual intervention—features absent in traditional deployments.
4. **Monitoring and Observability**: Legacy systems often rely on basic uptime monitoring. DevOps introduces real-time metrics, logs, and alerts, allowing proactive performance tuning and incident response.
5. **Security Integration**: Security is often bolted on at the end in older models. The DevOps approach integrates security early via **DevSecOps**, using tools for image scanning, secret management, and IAM policies.
6. **Team Collaboration**: Traditional models silo development and operations, causing delays and blame-shifting. DevOps encourages shared responsibility, reducing friction and increasing overall productivity.

VI. LIMITATIONS & DRAWBACKS

While the DevOps-based deployment of the E-commerce Robot Store offers numerous benefits such as automation, faster releases, and high availability, there are still certain limitations and drawbacks that must be acknowledged. Understanding these helps in setting realistic expectations and planning for future improvements:

1. High Initial Setup Complexity

Implementing a full DevOps pipeline with CI/CD, container orchestration (Kubernetes), and infrastructure as code (Terraform, Ansible) involves a steep learning curve and significant initial configuration effort. Teams must invest time in tool integration, pipeline scripting, and cloud infrastructure setup.

2. Resource-Intensive Operations

Running multiple microservices, containers, monitoring tools (like Prometheus and Grafana), and CI/CD servers consumes substantial computing resources. This can lead to increased costs, especially for small teams or startups operating on limited budgets.

3. Toolchain Fragmentation

DevOps relies on many tools (e.g., GitHub, Jenkins, Docker, Kubernetes, SonarQube, AWS, etc.), and managing the interoperability and updates of these tools can become complex. Tool misconfiguration or version conflicts can lead to deployment failures or inconsistencies.

4. Requires Skilled Personnel

Successful DevOps implementation demands skilled professionals who are well-versed in both development and operations. Lack of expertise in scripting, automation, Kubernetes, and cloud infrastructure can slow down the adoption process and increase the risk of errors.

5. Monitoring Overhead

While comprehensive monitoring is a strength of DevOps, it also introduces overhead. Continuous logging, metric collection, and alerting systems require tuning and maintenance. Improper configurations may result in alert fatigue or missed incidents.

6. Security Misconfigurations

Although DevSecOps practices are integrated, misconfigured IAM roles, open ports, insecure containers, or overlooked vulnerabilities can lead to security risks. The dynamic nature of DevOps environments may make it harder to maintain strict security controls consistently.

7. Microservices Management Complexity

Managing multiple microservices increases the complexity of deployment, service discovery, version control, and data consistency. Errors in one service may indirectly impact others if dependencies are not properly isolated.

8. Dependency on Cloud Providers

The system's performance and cost-efficiency rely heavily on cloud infrastructure (e.g., AWS). This creates a dependency on specific providers and may raise concerns around vendor lock-in and long-term cost management.

VII. RESULTS AND DISCUSSION

The implementation of the E-commerce Robot Store using DevOps practices yielded several important outcomes, demonstrating the effectiveness of modern software delivery pipelines, automation, and cloud-native technologies. This section presents both the **quantitative and qualitative results**, followed by a discussion on their implications.

1. Deployment Speed and Frequency

Result:

After implementing CI/CD pipelines using Jenkins, the team was able to perform multiple deployments per day compared to the previous cycle of bi-weekly or monthly releases. Deployment times were reduced from hours to under 5 minutes per service.

Discussion:

This improvement aligns with industry benchmarks and confirms that DevOps significantly reduces deployment bottlenecks. Frequent, small releases also minimized risks and made it easier to roll back problematic changes.

2. System Scalability and Uptime

Result:

Using Kubernetes on AWS EKS, the application demonstrated automatic horizontal scaling during peak traffic, maintaining an uptime of 99.95% over a 30-day monitoring period.

Discussion:

Scalability is essential for e-commerce platforms with unpredictable demand. Kubernetes' self-healing and auto-scaling capabilities played a crucial role in maintaining service availability during traffic surges.

3. Error Recovery and Rollbacks

Result:

The average Mean Time to Recovery (MTTR) was measured at less than 10 minutes, due to containerized services and version-controlled deployments.

Discussion:

This validates the DevOps claim of faster recovery through rollback mechanisms, automated testing, and immutable infrastructure. As a result, the platform ensured minimal downtime and quick customer-facing fixes.

© 2025 IJMRSET | Volume 8, Issue 6, June 2025| DOI:10.15680/IJMRSET.2025.0806162

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

4. Infrastructure Management Efficiency
Result:
Terraform and Ansible allowed provisioning of cloud infrastructure in under 20 minutes, with consistent and reproducible environments across dev, staging, and production.

Discussion:
Infrastructure as Code (IaC) dramatically reduced manual errors and environment drift. This consistency across environments contributed to smoother deployments and fewer post-release issues.

5. Monitoring and Observability
Result:
The integrated Prometheus-Grafana stack offered real-time visibility into CPU, memory, and service health. Alerts triggered for anomalies helped resolve issues proactively.

Discussion:
Better observability enabled faster decision-making and improved operational resilience. Proactive monitoring reduced incident impact and ensured performance compliance with SLAs.

6. User Experience and Feedback
Result:
Post-deployment surveys and analytics showed a 20% increase in page load speed, 15% reduction in checkout drop-off rates, and improved customer satisfaction scores.

Discussion:
These improvements are directly attributed to faster backend responses, optimized frontend delivery via CDNs, and fewer interruptions due to downtime or bugs. DevOps practices thus enhanced both backend efficiency and end-user satisfaction.

7. Security and Compliance
Result:
Automated vulnerability scans (e.g., Trivy, SonarQube) helped identify and fix critical security issues early in the pipeline. Role-based access and encrypted secrets were effectively managed using AWS IAM and Secrets Manager.

Discussion:
By shifting security left (DevSecOps), the system proactively addressed risks before production, demonstrating compliance readiness and improving trustworthiness.

VIII. CONCLUSION

The **E-commerce Robot Store** is an innovative and user-centric platform designed to revolutionize the shopping experience for customers by offering a seamless, intuitive, and secure online environment for purchasing robotic products. This platform efficiently handles various critical functions, including product catalog management, secure user registration and authentication, shopping cart management, order processing, payment gateway integration, and delivery tracking.

By automating these key processes, the store significantly reduces operational overhead, ensures a smooth customer experience, and enhances order accuracy and fulfillment efficiency. Customers enjoy an easy-to-navigate interface for browsing, selecting, and purchasing robotic products, while the backend system allows efficient order management, inventory control, and real-time analytics.

In summary, the **E-commerce Robot Store** not only provides a reliable and engaging platform for purchasing robotic products but also lays the groundwork for future expansions, such as AI-powered recommendations, product customization, and smart robotics integration, ensuring continuous growth and improved customer satisfaction in the long term.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

REFERENCES

1. “E-Commerce 2025: Business, Technology, Society”: Kenneth C. Laudon & Carol Guercio Traver
2. “Building E-Commerce Websites: A Complete Guide for Beginners”: David C. Layton
3. “Digital Commerce 360: E-Commerce and Business Strategy”: D. J. Scott.
4. “Design and Implementation of E-Commerce Website for Retail Store”: A. Sharma, P. Verma.
5. “Impact of AI in E-Commerce and Retail”: J. Liu, Y. Zhang