IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15371

Secure Web Application Using Docker, Nginx, and AWS WAF

Jagatheeshwaran E[1], Dr. C. Daniel Nesakumar[2]

III BCA, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous),

Coimbatore – 641006, Tamil Nadu, India[1]

Assistant Professor, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous),

Coimbatore – 641006, Tamil Nadu, India[2]

Abstract: The rapid expansion of internet-based services has increased the demand for secure, scalable, and reliable web application deployment environments. Traditional deployment methods often face challenges such as inconsistent environments, security vulnerabilities, and difficulty managing application scalability. This project presents the design and implementation of a secure web application deployment environment using Docker, Nginx, and AWS Web Application Firewall (WAF). Docker provides containerization to ensure application portability and consistency across development, testing, and production environments. Nginx acts as a reverse proxy server handling request routing, load balancing, and security filtering. AWS WAF monitors and blocks malicious HTTP/HTTPS traffic including SQL injection and cross-site scripting (XSS) attacks. The infrastructure is hosted on Amazon EC2 instances, providing scalability and high availability. The proposed system improves security, simplifies deployment, and ensures consistent performance across different environments.

Keywords: Docker, Containerization, Nginx, Reverse Proxy, AWS WAF, Web Security, Amazon EC2, Cloud Computing, SQL Injection, XSS Protection.

I. INTRODUCTION

The rapid growth of web applications has increased the demand for secure, scalable, and reliable deployment environments. Traditional deployment methods often face challenges such as inconsistent environments, security vulnerabilities, and difficulty in managing application scalability. To address these issues, modern technologies such as containerization, reverse proxy servers, and cloud-based security services are widely adopted.

Docker plays a major role in this project by providing containerization technology. Containerization allows the application and all its dependencies to be packaged together in a single container, ensuring the application runs consistently across different environments. By using Docker containers, the deployment process becomes faster, more reliable, and easier to manage. Containers also reduce system resource usage compared to traditional virtual machines.

Nginx is used as a reverse proxy server and web server. It acts as an intermediary between the client and the backend application containers, providing request routing, load balancing, traffic management, caching, and improved security. To enhance the security of the web application, AWS Web Application Firewall (WAF) is integrated into the system. AWS WAF monitors incoming HTTP and HTTPS requests, filtering malicious traffic using predefined and custom rules to detect and block SQL injection, XSS, and other malicious activities.

The entire infrastructure is deployed on Amazon EC2 instances, providing a flexible and scalable cloud computing environment. This project demonstrates how modern cloud technologies can be combined to create a secure and efficient web application deployment architecture that improves security, simplifies application deployment, and ensures consistent performance.

II. RELATED WORK

Several studies have explored the use of cloud computing and containerization for secure application deployment. Traditional server-based deployment systems require significant manual configuration and are difficult to scale as data volumes increase. Organizations using on-premise infrastructure face challenges such as hardware failures, capacity planning, and high operational costs.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.15371

Research on containerization has demonstrated that Docker significantly reduces infrastructure overhead while improving application portability and isolation. The concept of cloud computing as a shift from owned resources to utility-based computing established the theoretical foundation for modern containerized architectures. Studies have shown that container-based deployments improve system efficiency and reduce operational costs compared to traditional approaches.

Nginx has been widely recognized as a high-performance reverse proxy solution for managing web traffic. Its support for load balancing, SSL termination, and request filtering makes it an ideal intermediary layer in secure deployment architectures. AWS WAF has similarly been recognized as a reliable cloud-based security service capable of protecting web applications from common OWASP Top 10 threats without requiring manual security engineering from application developers.

III. EXISTING SYSTEM AND DRAWBACKS

In many organizations, web applications are deployed directly on physical servers or basic virtual machines without containerization, traffic management, or cloud-based security mechanisms. These traditional systems, while capable of basic hosting, suffer from several significant limitations.

The lack of application isolation is a major drawback. Multiple applications running on the same server share system libraries and configurations, creating compatibility issues and dependency conflicts. Without proper isolation, a failure in one application component can affect other services on the same host.

Security is another significant concern. Traditional web servers may not have strong protection against SQL injection, XSS, distributed denial-of-service (DDoS), or malicious request flooding. Many systems rely only on basic firewall configurations, which are insufficient to detect and block sophisticated application-layer attacks. Additionally, traditional deployment environments lack efficient traffic management and load balancing, leading to performance bottlenecks during high-traffic conditions.

Scalability presents further challenges. Adding capacity requires manual server configuration and hardware procurement, which is time-consuming and may require service downtime. Maintaining consistent environments across development, testing, and production stages also introduces significant configuration overhead and the risk of environment-specific errors.

IV. OBJECTIVES

The primary objectives of this project are as follows:

- Design and implement a secure web application deployment environment using Docker, Nginx, and AWS WAF.
- Use Docker containerization to ensure application portability and consistency across all environments.
- Configure Nginx as a reverse proxy server for efficient request routing, load balancing, and connection management.
- Integrate AWS WAF to protect the application from SQL injection, XSS, and other common web-based attacks.
- Deploy the complete infrastructure on Amazon EC2 for scalable and highly available cloud hosting.
- Implement monitoring and logging mechanisms using server logs and AWS WAF traffic analytics.

V. SYSTEM ARCHITECTURE

The proposed system follows a layered security architecture designed for scalability, reliability, and protection against common cyber threats. Each layer serves a distinct purpose and communicates with adjacent layers through well-defined interfaces.

The architecture consists of five primary layers. User requests originating from web browsers first reach the AWS WAF layer, which inspects all incoming HTTP and HTTPS traffic against predefined security rules. Requests that pass security checks are forwarded to the Amazon EC2 instance. Within EC2, the Nginx reverse proxy receives the request and routes it to the appropriate Docker container hosting the web application. The containerized application processes the request and returns the response through the same path back to the user.

The layered structure provides defense-in-depth: malicious traffic is blocked at the WAF layer before it can reach the application, while Nginx provides an additional intermediary that hides internal server details and manages connections efficiently.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15371

Layer	Technology	Function
Firewall Layer	AWS WAF	Inspect and block malicious requests
Cloud Infrastructure	Amazon EC2	Host Docker containers and Nginx
Reverse Proxy	Nginx	Request routing and load balancing
Application Layer	Docker Container	Run web application in isolation
Monitoring	Server Logs / AWS WAF Logs	Track traffic and security events

Table 1. System Architecture Layers

VI. IMPLEMENTATION

A. EC2 Instance Configuration

An Amazon EC2 instance running Ubuntu Linux was launched to serve as the hosting environment. Security groups were configured to allow inbound traffic on Port 22 (SSH), Port 80 (HTTP), and Port 443 (HTTPS). The instance provides the computing resources required to run Docker containers and the Nginx server.

B. Docker Containerization

Docker was installed on the EC2 instance to enable containerized application deployment. A Python Flask web application was packaged into a Docker container using a Dockerfile that specifies the base image, dependencies, and startup commands. The application runs inside an isolated container environment, ensuring consistent behavior across all deployment stages.

C. Nginx Reverse Proxy Configuration

Nginx was installed and configured as a reverse proxy server. The nginx.conf configuration file defines how incoming requests are routed to the application container. The reverse proxy listens on port 80 and forwards requests to the Flask application running inside the Docker container. This configuration hides internal server details from external users and improves request handling efficiency.

D. AWS WAF Configuration

A Web Access Control List (Web ACL) was created in AWS WAF and associated with the EC2 deployment. Multiple AWS Managed Rule Groups were enabled to provide comprehensive protection, as detailed in Table 2. Rule priority was configured to ensure that IP-based rules are evaluated before application-layer rules.

Rule Group	Protection Provided	Action
Core Rule Set	OWASP Top 10 attacks	Block
SQL Database Rule	SQL injection attempts	Block
Known Bad Inputs	Malicious request patterns	Block
Linux OS Rule	Linux-specific attack vectors	Block
Anonymous IP List	VPN, Tor, and proxy traffic	Block

Table 2. AWS WAF Managed Rule Groups

E. Security Group Rules

Type	Protocol	Port	Description
SSH	TCP	22	Remote server login
HTTP	TCP	80	Standard web traffic
HTTPS	TCP	443	Encrypted web traffic

Table 3. Security Group Configuration

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15371

VII. DESCRIPTION OF MODULES

1. User Request Module: Handles requests from users accessing the web application through a browser. User requests include URL parameters, HTTP headers, cookies, and form data, all of which are forwarded to the security inspection layer.

2. AWS WAF Security Module: Inspects incoming HTTP and HTTPS requests using a Web ACL containing security rules. Requests matching malicious patterns (SQL injection, XSS, bot traffic, IP-based attacks) are blocked before reaching the application server.

3. Reverse Proxy Module (Nginx): Routes incoming web traffic to the correct backend container. Manages server connections, handles multiple client requests efficiently, provides load balancing, and hides internal server details from external users.

4. Docker Container Module: Runs the web application inside isolated containers. Ensures environment consistency, simplifies deployment, and provides application isolation so that container failures do not affect other services.

5. Cloud Infrastructure Module (EC2): Provides the virtual computing environment hosting Docker containers and the Nginx server. Manages network configurations, security groups, and resource allocation for scalable application hosting.

6. Monitoring and Logging Module: Tracks system performance, traffic activity, and security events. Nginx access and error logs record incoming requests and server issues, while AWS WAF logs record blocked requests and attack patterns.

VIII. TESTING AND EVALUATION

The system was evaluated through multiple testing methodologies to ensure correctness, security, and performance.
Unit Testing: Each system component was tested independently. Docker container creation and execution were verified, Nginx configuration files were validated, and AWS WAF rule functionality was confirmed.

Integration Testing: The complete request flow was tested — from browser request through AWS WAF inspection, EC2 routing, Nginx forwarding, Docker container processing, and response delivery back to the client.

Security Testing: SQL injection and XSS attack payloads were sent as test requests. AWS WAF successfully detected and blocked all malicious test inputs before they reached the Nginx or application layers.

Performance Testing: The system was tested under simulated high-traffic conditions. Nginx efficiently managed multiple concurrent connections with consistent response times.

Component	Test Type	Result
Docker Container	Unit / Integration	Application runs consistently across environments
Nginx Reverse Proxy	Integration / Performance	Requests routed correctly; stable under load
AWS WAF	Security	SQL injection and XSS payloads blocked successfully
EC2 Infrastructure	Integration	Accessible via public IP; scalable deployment confirmed
End-to-End Workflow	System Testing	Full request-response cycle completed without errors

Table 4. Testing and Evaluation Results

IX. CONCLUSION

This project successfully designed and implemented a secure web application deployment environment by integrating Docker, Nginx, and AWS WAF on Amazon EC2 cloud infrastructure. The proposed layered architecture provides defense-in-depth against common cyber threats while ensuring application portability, consistent deployment, and efficient traffic management.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.15371

Docker containerization eliminates environment inconsistencies and reduces deployment complexity. Nginx provides efficient reverse proxy functionality with load balancing support and an additional security layer that hides internal server details. AWS WAF delivers cloud-based protection against OWASP Top 10 threats without requiring manual security engineering from application developers. Amazon EC2 provides scalable, highly available cloud infrastructure suitable for production workloads.

The evaluation results confirm that the system successfully blocks malicious requests, handles concurrent user traffic efficiently, and maintains consistent application behavior across deployment stages. This architecture represents a practical, modern approach to securing web applications in cloud environments, applicable to organizations seeking to replace traditional vulnerable deployment models.

X. FUTURE ENHANCEMENTS

Several enhancements can further extend the capabilities of the proposed system:

- Container orchestration using Kubernetes to automate container scaling, load balancing, and lifecycle management.
- CI/CD pipeline integration using Jenkins or GitHub Actions for automated build, test, and deployment workflows.
- Advanced monitoring using Prometheus and Grafana for real-time performance dashboards and alerting.
- DDoS protection using AWS Shield and improved access control using AWS IAM roles.
- HTTPS encryption using SSL/TLS certificates from Let's Encrypt for secured client-server communication.
- Microservices architecture decomposing the application into independent services for improved scalability.
- Automated vulnerability scanning tools integrated into the deployment pipeline for continuous security assessment.

REFERENCES

[1]. Docker Inc., "Docker Documentation," https://docs.docker.com, 2024.
[2]. Nginx Inc., "Nginx Official Documentation," https://nginx.org/en/docs/, 2024.
[3]. Amazon Web Services, "AWS WAF Developer Guide," https://docs.aws.amazon.com/waf/, 2024.
[4]. Amazon Web Services, "Amazon EC2 User Guide," https://docs.aws.amazon.com/ec2/, 2024.
[5]. OWASP Foundation, "OWASP Top 10 Web Application Security Risks," https://owasp.org/Top10/, 2021.
[6]. Armbrust, M. et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.
[7]. Merkel, D., "Docker: Lightweight Linux Containers for Consistent Development and Deployment," Linux Journal, vol. 239, 2014.
[8]. Amazon Web Services, "AWS Security Best Practices," AWS Whitepaper, 2024.

