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.2026.15302

Dockerized Microservices Architecture on AWS ECS

M. Sathya Narayanan[1], Dr. B. Narasimhan[2]

III BCA, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore, Tamil Nadu, India.[1]

Assistant Professor, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore, Tamil Nadu, India.[2]

Abstract: Modern software systems require flexible and scalable deployment environments to support growing workloads and complex applications. Containerization has emerged as an efficient solution that allows applications to be packaged with all required dependencies into lightweight containers. Docker is one of the most widely used containerization technologies that enables developers to build portable and consistent application environments. However, managing containers in large-scale environments requires a reliable orchestration platform.

This research presents a cloud-based deployment architecture titled **"Dockerized Microservices Architecture on AWS ECS."** The system integrates Docker containerization with several Amazon Web Services (AWS) components including Amazon Elastic Container Service (ECS), Amazon Elastic Container Registry (ECR), Amazon Elastic Compute Cloud (EC2), and AWS Identity and Access Management (IAM).

Docker is used to build container images for the application, while Amazon ECR stores these images securely in private repositories. Amazon ECS is responsible for container orchestration, enabling automated deployment, monitoring, and scaling of containerized applications. Amazon EC2 provides the computing infrastructure required to run containers within the ECS cluster. IAM ensures secure authentication and role-based access control for cloud resources.

The proposed system demonstrates a scalable and secure container deployment model that improves application portability, deployment efficiency, and resource utilization. The results show that combining Docker with AWS ECS can significantly simplify cloud deployment and support modern DevOps practices for building reliable microservices-based applications.

Keywords: Docker, Microservices Architecture, AWS ECS, Cloud Deployment, Containerization, DevOps.

I. INTRODUCTION

Cloud computing has become a fundamental technology for modern software development and deployment. Organizations increasingly rely on cloud platforms to host applications because they offer scalability, reliability, and flexible resource management. However, traditional application deployment methods often depend on monolithic architectures and manual server configuration. These approaches can lead to deployment inconsistencies, system downtime, and complex maintenance processes.

Containerization has emerged as an effective approach to address these challenges. Containers allow applications to run in isolated environments while sharing the underlying operating system. This ensures that applications behave consistently across development, testing, and production environments. Docker is one of the most widely adopted container platforms used to build, package, and distribute containerized applications.

Although Docker simplifies application packaging, managing multiple containers across servers requires orchestration tools. Container orchestration platforms automate deployment, scaling, and monitoring of containers in distributed environments. Amazon Web Services provides **Amazon Elastic Container Service (ECS)**, a fully managed container orchestration platform designed to run and manage Docker containers in the cloud.

This research focuses on implementing a containerized microservices deployment system using Docker and AWS ECS. The architecture integrates several AWS services including IAM for security, ECR for container image storage, ECS for container orchestration, and EC2 for infrastructure management. The goal of the project is to demonstrate a

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.2026.15302

complete container deployment lifecycle in a cloud environment while ensuring scalability, security, and operational efficiency.

II. RELATED WORK

Several studies have explored the use of containerization and cloud orchestration technologies to improve application deployment processes. Traditional deployment methods rely on installing applications directly on physical servers or virtual machines. These systems often require manual configuration of dependencies and runtime environments, which can lead to compatibility issues and deployment errors.

Container technologies such as Docker have been introduced to solve these problems by providing lightweight and portable environments for application execution. Docker containers package application code together with its dependencies, ensuring that the application runs consistently across different environments.

In recent years, container orchestration platforms such as Kubernetes, Docker Swarm, and Amazon ECS have been developed to manage large-scale container deployments. These platforms automate tasks such as container scheduling, health monitoring, load balancing, and scaling.

Among these orchestration tools, Amazon ECS offers strong integration with other AWS services, making it suitable for organizations already using AWS cloud infrastructure. ECS simplifies container deployment by providing managed clusters and automated service management. By combining containerization with cloud orchestration, organizations can build scalable and reliable cloud-native applications.

III. OBJECTIVES AND CHALLENGES

Objectives

The primary objectives of this project are:

1. To implement application containerization using Docker.
2. To design a microservices deployment architecture in a cloud environment.
3. To store Docker images securely using Amazon Elastic Container Registry (ECR).
4. To deploy containerized applications using Amazon Elastic Container Service (ECS).
5. To implement secure access control using AWS Identity and Access Management (IAM).
6. To demonstrate scalable and automated container deployment using AWS infrastructure.

Development Challenges

During the implementation of the system, several challenges were encountered while integrating Docker with AWS cloud services. One of the major challenges involved configuring IAM policies to ensure secure access control while allowing required permissions for container deployment.

Another challenge involved authenticating Docker with Amazon ECR and pushing container images securely to the repository. Correct configuration of AWS CLI credentials and repository authentication was necessary to ensure successful image uploads.

Managing ECS task definitions and cluster configurations also required careful planning. Resource allocation such as CPU and memory limits had to be configured properly to ensure stable container performance. Additionally, troubleshooting deployment errors and resolving container startup issues were important tasks during system implementation.

IV. SYSTEM ARCHITECTURE

The proposed system architecture follows a cloud-based container deployment model that integrates Docker containerization with AWS cloud services. The architecture consists of multiple layers that work together to build, store, and deploy containerized applications.

The **Application Layer** represents the web application that is prepared for containerization. The application code is packaged using Docker along with its dependencies.

The **Containerization Layer** uses Docker to build container images. A Dockerfile is used to define the base

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.2026.15302

environment, application files, and startup commands required to run the container.

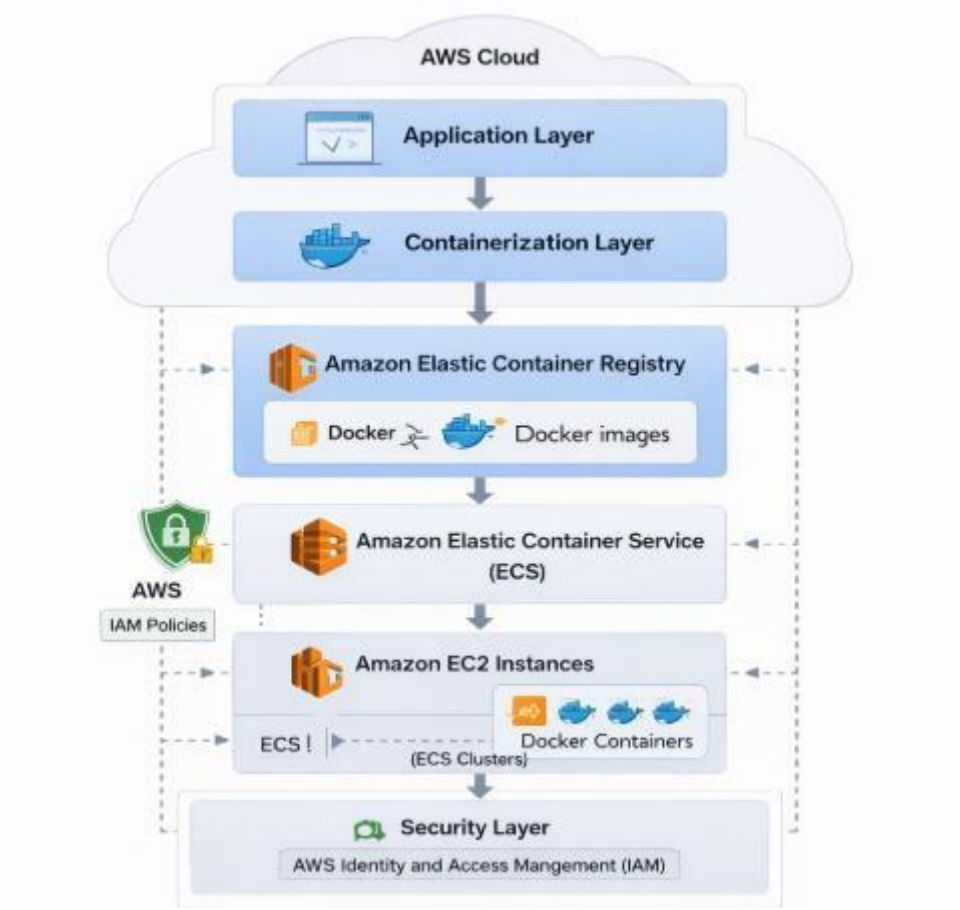
The **Image Repository Layer** is handled by Amazon Elastic Container Registry (ECR). This service stores Docker images securely in private repositories and allows ECS to retrieve the required images during deployment.

The **Orchestration Layer** is managed by Amazon Elastic Container Service (ECS). ECS is responsible for deploying containers, managing clusters, maintaining desired task counts, and monitoring container health.

The **Infrastructure Layer** is provided by Amazon EC2 instances. These instances host Docker containers and provide the computing resources needed for container execution.

Security is implemented through **AWS Identity and Access Management (IAM)**, which controls access permissions and ensures secure communication between AWS services.

This architecture enables automated deployment, improved scalability, and efficient management of containerized applications.



V. IMPLEMENTATION

The system implementation involved several steps starting from container creation to final cloud deployment.
Initially, the application was containerized using Docker. A Dockerfile was created to define the container environment and copy application files into the container image. The Docker image was then built using Docker commands.
After building the container image, it was pushed to Amazon Elastic Container Registry (ECR). This repository stores Docker images securely and allows ECS to retrieve them during deployment.
Next, an ECS cluster was created to manage container instances. ECS task definitions were configured to specify container settings such as image location, CPU allocation, memory limits, and network configurations.
Finally, the ECS service was launched to deploy containers within the cluster. The application was successfully

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.2026.15302

accessed through the public IP address of the EC2 instance.
The implementation demonstrated how containerized applications can be deployed efficiently using AWS cloud infrastructure.

VI. EVALUATION RESULTS AND DISCUSSIONS

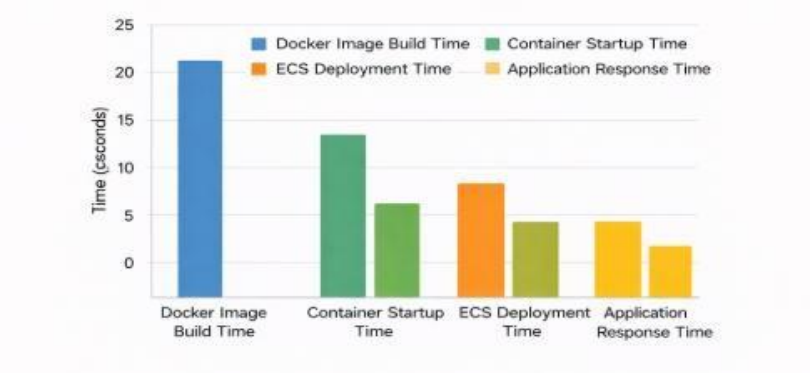
The system was evaluated based on deployment performance, container startup time, and application accessibility. Several tests were conducted to measure the efficiency of the container deployment process.

Performance evaluation included metrics such as Docker image build time, container startup time, ECS deployment duration, and application response time. The results showed that the containerized deployment model significantly reduces deployment complexity and improves system reliability.

Containers started quickly after deployment and ECS automatically maintained the required number of running tasks. The application was accessible through the public IP address without significant delay, indicating successful deployment.

The system also demonstrated efficient resource utilization because containers share the host operating system instead of running separate virtual machines. This reduces infrastructure overhead and improves overall system performance.

Overall, the evaluation results confirm that Docker-based microservices deployment using AWS ECS provides a reliable and scalable solution for modern cloud applications.



VII. CONCLUSION

This research successfully implemented a Dockerized microservices architecture using Amazon Web Services. The integration of Docker with AWS services such as ECS, ECR, EC2, and IAM provides a secure and scalable platform for deploying containerized applications.

Docker simplifies application packaging and ensures consistency across environments, while Amazon ECS automates container orchestration and service management. The use of IAM enhances system security through controlled access permissions and role-based authentication.

The proposed architecture replaces traditional monolithic deployment models with a cloud- native container-based system. This approach improves scalability, reduces operational complexity, and supports modern DevOps practices.
Overall, the project demonstrates how containerization and cloud orchestration technologies can be combined to create efficient and reliable application deployment frameworks.

VIII. FUTURE ENHANCEMENTS

Several improvements can be implemented to enhance the system in the future.

One possible enhancement is the integration of **auto-scaling mechanisms** to automatically adjust container instances based on workload demand. This will improve system performance during high traffic conditions.

DOI: 10.17148/IJARCCE.2026.15302

Another improvement is the integration of **load balancers**, such as AWS Application Load Balancer, to distribute incoming traffic across multiple container instances and improve system availability.

The system can also be extended by implementing **CI/CD pipelines** using tools such as GitHub Actions or AWS CodePipeline. This would automate the process of building, testing, and deploying container images.

Additional enhancements may include multi-region deployment, advanced monitoring using Amazon CloudWatch, and infrastructure provisioning using Infrastructure as Code tools such as AWS CloudFormation or Terraform.

REFERENCES

[1]. Amazon Web Services Documentation – https://docs.aws.amazon.com
[2]. Docker Documentation – https://docs.docker.com
[3]. Merkel, D. "Docker: Lightweight Linux Containers for Consistent Development and Deployment."
[4]. Wittig, A., & Wittig, M. *Amazon Web Services in Action.*
[5]. Turnbull, J. *The Docker Book: Containerization is the New Virtualization.*
[6]. IEEE Cloud Computing Journal – Microservices Architecture Research Papers.
[7]. Cloud Security Alliance – Best Practices for Secure Cloud Deployment.

