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.2026.15385

# Implement Auto Scaling with Docker and AWS EC2 Auto Scaling

**Nishok T N[1], Dr. S. Thavamani[2]**

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

Coimbatore – 641006, Tamil Nadu, India[1]

Associate Professor, Department of Computer Applications,

Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore – 641006, Tamil Nadu, India[2]

**Abstract:** Cloud computing has become an essential technology for modern software development and deployment. One of the key advantages of cloud computing is the ability to scale resources according to application demand. In traditional systems, servers are configured with fixed resources, which can cause performance problems when user traffic increases and resource wastage when traffic decreases. Auto scaling is a cloud computing feature that helps solve this issue by automatically increasing or decreasing computing resources based on workload.

This project focuses on implementing an auto scaling infrastructure using Docker containers and AWS EC2 instances. Docker is a containerization platform that allows developers to package applications and their dependencies into lightweight containers. These containers ensure that the application runs consistently across different environments. AWS EC2 provides virtual machines in the cloud that can host these containers.

In the proposed architecture, an Elastic Load Balancer distributes incoming traffic across multiple EC2 instances. Each instance runs a Docker container that hosts the application. AWS Auto Scaling monitors system metrics such as CPU utilization and automatically launches additional instances when the load increases. Similarly, when the load decreases, extra instances are terminated to optimize resource usage and reduce cost.

This system demonstrates how containerization and cloud auto scaling technologies can work together to create scalable, reliable, and cost-efficient cloud applications.

## 1. INTRODUCTION

The rapid growth of internet services has created the need for highly scalable and reliable application infrastructures. Modern applications must be capable of handling sudden increases in traffic without affecting performance or user experience. Traditional server infrastructure requires manual scaling, which is time consuming and inefficient. In many cases, organizations either over-provision resources to handle peak loads or suffer performance issues when demand exceeds available capacity.

Cloud computing has introduced several solutions to these challenges. One of the most important features offered by cloud platforms is auto scaling. Auto scaling automatically adjusts computing resources based on application demand. This ensures that the system always has the right amount of resources available.

Containerization has also become an important technology for application deployment. Docker is one of the most widely used container platforms. It allows developers to package applications along with their dependencies into portable containers that can run consistently across different systems.

This project integrates Docker containerization with AWS EC2 auto scaling to create a scalable architecture. By combining these technologies, applications can automatically respond to changing workloads while maintaining high availability and performance.

## 2. OBJECTIVES OF THE PROJECT

The main objectives of this project are listed below.

First, the project aims to understand the fundamental concepts of cloud computing and scalable system architecture. Cloud services provide flexible computing resources that can be adjusted according to user demand.

Second, the project focuses on learning containerization technology using Docker. Containers provide lightweight virtualization and enable faster application deployment.

Third, the project aims to deploy an application inside a Docker container and run it on an AWS EC2 instance. This helps demonstrate how containers can be used in real cloud environments.

Fourth, the project aims to configure AWS Auto Scaling groups to automatically manage EC2 instances based on workload metrics.

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.2026.15385

Fifth, the project aims to implement a load balancing mechanism to distribute user requests among multiple instances. Finally, the project demonstrates how all these components work together to build a reliable and scalable cloud application system.

## 3. TOOLS AND TECHNOLOGIES USED

Several tools and technologies were used in this project.
Amazon Web Services (AWS):
AWS is one of the leading cloud computing platforms. It provides a wide range of services including computing power, storage, networking, and monitoring tools. AWS enables developers to deploy applications quickly and scale them according to demand.
Amazon EC2:
Amazon Elastic Compute Cloud (EC2) provides virtual servers in the cloud. These servers can run applications just like physical machines but with the advantage of flexible configuration and scalability.
Docker:
Docker is a containerization platform that allows applications to run inside containers. Containers package the application code, libraries, and dependencies together, ensuring that the application runs consistently across environments.
AWS Auto Scaling:
AWS Auto Scaling automatically adjusts the number of EC2 instances based on system demand. It monitors metrics such as CPU utilization and launches or terminates instances accordingly.
Elastic Load Balancer:
Elastic Load Balancer distributes incoming network traffic among multiple EC2 instances. This ensures high availability and prevents any single instance from becoming overloaded.
Visual Studio Code:
Visual Studio Code is used as the development environment for writing and managing application code.

## 4. SYSTEM ARCHITECTURE

The system architecture consists of multiple cloud components working together to provide scalability and reliability.
Users access the application through the internet. Their requests are first received by the Elastic Load Balancer. The load balancer distributes these requests across multiple EC2 instances.
Each EC2 instance runs a Docker container that hosts the application. Docker ensures that the application environment remains consistent across all instances.
AWS Auto Scaling continuously monitors system metrics such as CPU utilization. When the CPU usage exceeds a predefined threshold, the auto scaling service automatically launches additional EC2 instances using a predefined machine image. These instances are then registered with the load balancer.
When the application load decreases, the auto scaling service terminates unnecessary instances. This ensures efficient resource utilization and reduces operational cost.
The architecture therefore provides scalability, fault tolerance, and improved application performance.

## 5. IMPLEMENTATION PROCESS

The implementation of this project involves several important steps.
Step 1: Launch EC2 Instance
The first step is to launch an EC2 instance from the AWS management console. The instance acts as the base server where Docker will be installed.
Step 2: Install Docker
Docker is installed on the EC2 instance using Linux commands. After installation, the Docker service is started and verified.
Step 3: Deploy Application Container
The application is packaged into a Docker container image. This image contains all required dependencies and configuration files needed to run the application.
Step 4: Create Amazon Machine Image (AMI)
After configuring the EC2 instance and Docker environment, an Amazon Machine Image is created. This image serves as a template for launching additional instances.
Step 5: Configure Auto Scaling Group
An Auto Scaling Group is created using the AMI. Minimum, maximum, and desired instance counts are configured according to system requirements.

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.2026.15385

Step 6: Configure Load Balancer
An Elastic Load Balancer is configured and connected to the Auto Scaling Group so that incoming traffic can be distributed evenly among instances.

## 6. TESTING AND RESULTS

The implemented system was tested by simulating user traffic to the application. Artificial load was generated to observe how the system reacts to increasing demand.
During testing, when CPU utilization exceeded the predefined threshold, AWS Auto Scaling automatically launched additional EC2 instances. These instances were registered with the load balancer and started processing incoming requests.
As the number of user requests decreased, the Auto Scaling service gradually terminated extra instances. This ensured that only the necessary resources were used.
The results showed that the system was able to handle dynamic workloads efficiently. The combination of Docker containers and AWS Auto Scaling provided a reliable and flexible infrastructure capable of adapting to changing application demands.

## 7. ADVANTAGES OF THE SYSTEM

The proposed system offers several advantages.
One major advantage is scalability. The system can automatically increase computing resources when traffic increases.
Another advantage is high availability. Because multiple EC2 instances are running simultaneously, the application continues to function even if one instance fails.
Cost efficiency is another benefit. Resources are used only when needed, which reduces operational costs.
The system also improves deployment consistency because Docker containers ensure that applications run in the same environment across all servers.

## 8. CONCLUSION

This project successfully demonstrated the implementation of an auto scaling architecture using Docker and AWS EC2. The integration of containerization and cloud auto scaling technologies provides an efficient solution for modern application deployment.
Docker simplified application packaging and deployment by encapsulating the application and its dependencies into containers. AWS Auto Scaling ensured that the number of EC2 instances automatically adjusted according to system demand.
The final system improved application scalability, availability, and cost efficiency. In the future, this system can be enhanced by integrating monitoring dashboards, CI/CD pipelines, and advanced security mechanisms to further improve performance and reliability.

## REFERENCES

1. Amazon Web Services Documentation – https://docs.aws.amazon.com
2. Docker Official Documentation – https://docs.docker.com
3. Rajkumar Buyya, Cloud Computing: Principles and Paradigms.
4. AWS Auto Scaling User Guide.
5. Research articles and tutorials on Docker containerization and cloud scalability.