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.2026.15345

DOCKER SWARM CLUSTER ON AWS

A. Taniya Maria[1], Dr. C. Daniel Nesakumar[2]

III BCA, Department of Computer Applications, Sri Ramakrishna College of Arts and science (Autonomous), Coimbatore - 641006, Tamil Nadu, India[1]

Associate Professor, Department of Computer Applications, Sri Ramakrishna College of Arts and science (Autonomous), Coimbatore - 641006, Tamil Nadu, India[2]

Abstract: Docker Swarm is a native container orchestration platform that enables the deployment, management, and scaling of containerized applications across multiple machines. Implementing a Docker Swarm cluster on Amazon Web Services (AWS) provides a scalable, highly available, and fault-tolerant environment for modern cloud applications. In this architecture, multiple Amazon EC2 instances are configured as nodes within the swarm cluster, where manager nodes handle cluster management tasks such as scheduling, orchestration, and maintaining the desired state of services, while worker nodes run the application containers. AWS infrastructure components such as Elastic Compute Cloud (EC2), Elastic Load Balancer (ELB), Elastic Block Store (EBS), and Elastic File System (EFS) are integrated to support compute resources, traffic distribution, and persistent storage. The cluster is typically distributed across multiple availability zones to ensure high availability and resilience against failures. Docker Swarm simplifies container networking through overlay networks and provides built-in load balancing and service discovery. By deploying Docker Swarm on AWS, organizations can efficiently manage microservices-based applications, automate container deployment, and dynamically scale services according to workload demands while leveraging the reliability, security, and elasticity of the AWS cloud platform.

I. INTRODUCTION

One of the most popular container platforms is Docker. Docker makes it easier for developers to build, deploy, and manage applications inside containers. However, when applications need to run across multiple machines, it becomes difficult to manage many containers manually. In such cases, a container orchestration system is required to handle deployment, scaling, and management of containers.

Docker Swarm is Docker's built-in orchestration tool that allows multiple Docker hosts to form a cluster and work as a single system. It provides features such as load balancing, service discovery, scaling, and fault tolerance. In a swarm cluster, nodes are divided into manager nodes and worker nodes. Manager nodes control the cluster and schedule tasks, while worker nodes run the actual containers.

To deploy the swarm cluster in the cloud, this project uses Amazon Web Services. AWS provides scalable and reliable cloud infrastructure. Services like Amazon EC2 are used to create virtual machines that act as swarm nodes, while services such as Elastic Load Balancing and Amazon Elastic Block Store help manage traffic and storage.

The main objective of this project is to design and implement a Docker Swarm cluster on AWS for efficient deployment and management of containerized applications. Multiple EC2 instances are configured as manager and worker nodes, secure networking is established between them, and sample applications are deployed to demonstrate the functionality of the swarm cluster. This system helps improve application scalability, availability, and reliability.

II. SYSTEM STUDY

The system study for the Docker Swarm Cluster on AWS project focuses on understanding the technologies, tools, and infrastructure required to deploy and manage containerized applications in a cloud environment. With the rapid growth of cloud computing and microservices architecture, organizations require efficient methods to deploy applications that are scalable, portable, and easy to manage. Containerization has emerged as a solution to these challenges by allowing applications and their dependencies to run in isolated environments called containers.
One of the most widely used container platforms is Docker, which allows developers to package applications and their dependencies into containers. Docker ensures that applications run consistently across different systems such as local machines, servers, and cloud environments. However, when multiple containers are deployed across several machines,

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.2026.15345

managing them manually becomes complex. This creates the need for a container orchestration platform that can automate deployment, scaling, and management of containers.

To address this challenge, the project uses Docker Swarm, which is Docker's native clustering and orchestration tool. Docker Swarm enables multiple Docker hosts to work together as a single cluster. It manages container deployment, load balancing, service discovery, and fault tolerance. The swarm architecture consists of manager nodes that control cluster operations and worker nodes that run application containers, ensuring efficient distribution of workloads.

The project deploys the Docker Swarm cluster on the cloud using Amazon Web Services. AWS provides a flexible and scalable infrastructure for hosting applications. Virtual servers are created using Amazon EC2 to act as the nodes of the swarm cluster. This cloud-based setup allows the system to scale resources easily and ensures high availability of applications. Through this system study, the project analyzes the requirements, tools, and infrastructure necessary for implementing a Docker Swarm cluster on AWS.

III. SYSTEM DESIGN AND DEVELOPMENT

The system design for the Docker Swarm cluster on Amazon Web Services is based on a **manager-worker node architecture**, where manager nodes control cluster operations, maintain the desired state, and schedule services, while worker nodes run the actual application containers. The nodes communicate through **overlay networks**, ensuring secure and efficient data transfer between containers across multiple nodes. This architecture allows centralized management while distributing workloads effectively for better performance and reliability.

The cluster is deployed on AWS using **EC2 instances** for both manager and worker nodes. Leveraging cloud infrastructure provides the system with **scalability, high availability, and fault tolerance**, allowing applications to run efficiently even under heavy loads. **Docker Swarm** is used as the container orchestration platform, which automates the deployment, scheduling, and management of containers across the cluster. It also allows services to be **scaled up or down dynamically** according to demand, reducing manual intervention.

The networking design includes **secure communication** between nodes and proper configuration of AWS **security groups** and firewall rules to protect the cluster. **Persistent storage** is integrated using services like **Amazon EBS** or **Amazon EFS**, allowing containers to retain data even after restarts or migrations. Additionally, Docker Swarm provides **internal load balancing**, while AWS **Elastic Load Balancing (ELB)** can be used to distribute incoming traffic efficiently across nodes for optimal performance.

Monitoring and logging are important components of the system. Tools such as **Prometheus, Grafana, or AWS CloudWatch** are integrated to track container performance, node health, and system metrics. Logs are analyzed to detect failures or performance issues proactively. Deployment and scaling are automated using **Docker Swarm commands** or **Docker Compose**, ensuring consistent and rapid application deployment.

IV. SYSTEM DEVELOPMENT

The **system development** of a Docker Swarm cluster on Amazon Web Services involves designing, implementing, and testing a scalable, reliable, and secure environment for deploying containerized applications. The development process follows a structured approach to ensure efficiency and maintainability.

The development begins with **requirement analysis and planning**, where hardware and software specifications are determined, including the selection of EC2 instance types, storage options (EBS/EFS), and network configurations. The number of manager and worker nodes is decided based on the expected load and high availability requirements. Security considerations, such as firewall rules, secure authentication, and encrypted communication between nodes, are also included in the planning phase.

Implementation involves setting up the AWS infrastructure and installing Docker on all EC2 instances. The swarm cluster is initialized on the manager node using docker swarm init, and worker nodes are joined using the provided join tokens. Overlay networks are configured to enable secure inter-node communication, and persistent storage is attached to containers to maintain data consistency. Application services are defined using **Docker Compose or stack files** and deployed across the swarm cluster.

During development, **automation scripts** are created to simplify tasks such as node addition, service deployment, scaling, and updates. This reduces manual errors and ensures repeatable, reliable deployment processes. Monitoring and

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.2026.15345

logging tools like Prometheus, Grafana, and AWS CloudWatch are integrated to provide real-time performance metrics, alerts, and logs for cluster health and container status.

The **testing phase** ensures that all components work correctly. This includes verifying cluster formation, service deployment, load balancing, scaling operations, persistent storage access, and security configurations. Fault tolerance is tested by simulating node failures to ensure that services are automatically rescheduled and that high availability is maintained.

V. CONCLUSION

The project successfully demonstrated the deployment and management of a container orchestration environment using **Docker Swarm** on the cloud infrastructure provided by **Amazon Web Services**. By utilizing services such as **Amazon EC2**, the system was able to create a scalable cluster consisting of manager and worker nodes for running containerized applications.

The implementation showed how containers can be efficiently deployed, managed, and scaled across multiple nodes. The swarm cluster ensured high availability, load balancing, and fault tolerance, allowing services to continue running even when a node fails. Testing confirmed that the cluster could distribute workloads effectively, automatically reschedule tasks, and scale services according to demand.

Overall, integrating **Docker** containerization with cloud resources provides a flexible and reliable platform for modern application deployment. This project highlights the advantages of container orchestration in improving resource utilization, simplifying application management, and supporting scalable cloud-based solutions.

VI. FUTURE ENHANCEMNET

In the future, the Docker Swarm cluster on AWS can be enhanced by integrating auto scaling to automatically adjust resources based on workload. Monitoring and logging tools like AWS CloudWatch can be added to track system performance and detect issues quickly. Security improvements such as IAM roles, encrypted communication, and secure secret management can protect sensitive data. The system can also integrate CI/CD pipelines to automate application deployment. Additionally, load balancing and multi-region deployment can improve availability and reliability. These enhancements will make the system more scalable, secure, and efficient.

REFERENCES

[1]. Docker Documentation. (2024). Docker Swarm Mode Overview. Available at: https://docs.docker.com/engine/swarm/
[2]. Amazon Web Services Documentation. (2024). Amazon EC2 User Guide for Linux Instances. Available at: https://docs.aws.amazon.com/ec2/
[3]. Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal.
[4]. GitHub. (2024). Container Deployment and Version Control Documentation. Available at: https://github.com
[5]. Docker Swarm Documentation. Swarm Mode Key Concepts. Available at: https://docs.docker.com/engine/swarm/key-concepts/