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.2026.15348

SECURE AWS INFRASTRUCTURE WITH DOCKER AND IAM ROLES

Sananta.M[1], Dr. B. Narasimhan[2]

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 project, Secure AWS Infrastructure Using Docker and IAM Roles, focuses on implementing a secure and efficient infrastructure for deploying applications using services provided by Amazon Web Services. The application is deployed on a virtual server using Amazon EC2 and runs inside a containerized environment using Docker. Containerization ensures that the application and its dependencies are packaged together, enabling consistent execution and simplified deployment.

To enhance security, the project utilizes role-based access control through AWS Identity and Access Management. IAM roles allow the application running on the EC2 instance to securely access other AWS services without storing permanent credentials in the source code. This approach improves security by providing temporary and controlled permissions.

The system stores and retrieves application data using Amazon S3, which provides reliable and scalable storage. The application interacts with S3 using the AWS SDK, enabling secure communication between the application and storage service. Overall, the project demonstrates how containerization and role-based access control can be integrated to build a secure and manageable AWS infrastructure for modern application deployment.

Keywords: AWS Infrastructure, Docker, IAM Roles, Amazon EC2, Amazon S3, Containerization, AWS SDK.

I. INTRODUCTION

Modern application deployment requires secure, scalable, and reliable infrastructure to ensure efficient performance and data protection. Organizations increasingly use services from Amazon Web Services to deploy and manage applications because of their flexibility and wide range of cloud-based resources. However, ensuring secure access to these resources is an important aspect of infrastructure design. Traditional application deployment often involves installing applications directly on servers along with their dependencies. This approach can lead to compatibility issues and makes it difficult to maintain consistent environments. To overcome these challenges, containerization technologies such as Docker are widely used. Docker packages applications and their dependencies into containers, enabling them to run consistently across different systems.

Security is another major concern when applications interact with cloud services. Instead of storing sensitive access keys in the application code, secure access control mechanisms can be implemented using AWS Identity and Access Management. IAM roles provide temporary permissions that allow applications to securely access other AWS services without exposing credentials.

In this project, a secure infrastructure is implemented where a web application is deployed on Amazon EC2 and runs inside a Docker container. The application securely interacts with storage services such as Amazon S3 using IAM roles. This approach demonstrates how containerization and role-based access control can be combined to build a secure and manageable AWS infrastructure.

II. OBJECTIVES

- To implement a secure infrastructure on AWS:
  The project aims to build a secure application environment using services from Amazon Web Services, ensuring proper authentication and controlled access to resources.

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.2026.15348

- To deploy the application using containerization:
  The application is deployed using Docker, which packages the application and its dependencies into containers for consistent and reliable execution across different environments.

- To host the application on a virtual server:
  The project uses Amazon EC2 to provide a scalable and flexible computing environment for running the application.

- To implement secure access control using IAM roles:
  The project utilizes AWS Identity and Access Management roles to grant temporary permissions for accessing AWS services without storing sensitive credentials in the application code.

- To store and manage application data securely:
  The application stores and retrieves data using Amazon S3, which provides reliable and scalable storage.

- To demonstrate secure interaction between application and AWS services:
  The project demonstrates how applications can securely communicate with AWS services using IAM roles and secure APIs.

III. EXISTING SYSTEM

In traditional application deployment environments, applications are usually installed directly on physical servers or virtual machines. In this approach, the application and its dependencies are configured manually on the system. This process often leads to compatibility issues, configuration errors, and difficulties in maintaining consistent environments across multiple systems. Another major limitation of the existing system is related to security. Many applications store permanent access keys or credentials directly in the application code to access external services. If the source code is exposed or compromised, attackers can gain unauthorized access to system resources. In addition, managing infrastructure manually increases operational complexity and makes the deployment process time-consuming. Scaling applications in such environments is also difficult because resources must be configured manually due to these limitations, traditional deployment methods are less secure, less flexible, and harder to maintain.

IV. PROPOSED SYSTEM

The proposed system introduces a secure and modern infrastructure using services from Amazon Web Services combined with containerization and role-based access control. In this system, the application is deployed on a virtual server using Amazon EC2 and runs inside a container created using Docker. Docker packages the application and its dependencies together, ensuring that the application runs consistently across different environments. To enhance security, the system uses role-based access control through AWS Identity and Access Management. IAM roles allow the application to access AWS services securely without storing permanent credentials in the application code. The application data is stored and managed using Amazon S3. The application communicates with S3 through secure APIs using the AWS SDK. This approach provides a more secure, scalable, and efficient infrastructure compared to traditional deployment methods.

V. SYSTEM ARCHITECTURE

The system architecture of the project Secure AWS Infrastructure Using Docker and IAM Roles is designed to provide a secure, scalable, and efficient environment for deploying a web application. The architecture integrates containerization, virtual computing resources, secure authentication, and cloud storage services to ensure reliable application performance and controlled access to resources.

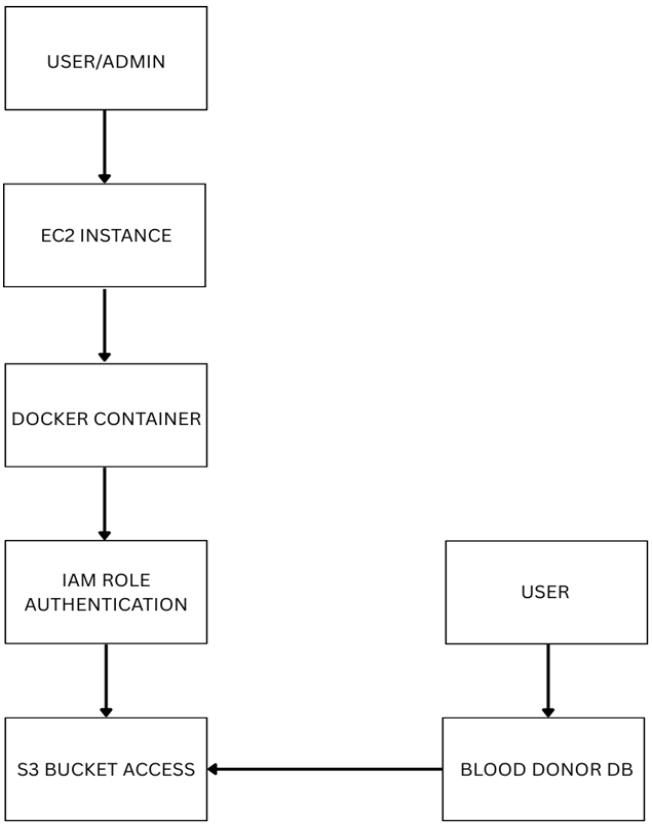
The application is hosted on a virtual server provided by Amazon EC2, which acts as the main computing environment. The EC2 instance runs a Docker container that contains the web application along with all required dependencies. Using Docker ensures that the application environment remains isolated and consistent, regardless of where the container is deployed. To maintain security, the EC2 instance is assigned an IAM role through AWS Identity and Access Management. This role provides the necessary permissions for the application to access specific AWS services without requiring permanent credentials in the application code. The use of IAM roles helps prevent credential exposure and ensures secure authentication.

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

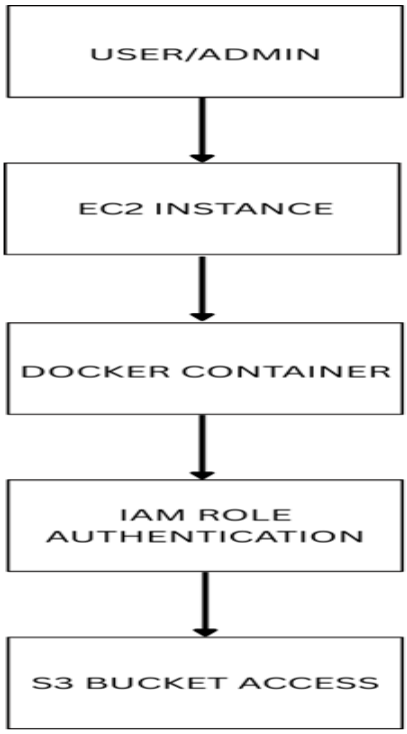
International Journal of Advanced Research in Computer and Communication Engineering

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

DOI: 10.17148/IJARCCE.2026.15348



The application stores and retrieves its data using Amazon S3, which provides highly reliable and scalable storage. The communication between the application and the S3 bucket is handled through API calls using the AWS SDK. The IAM role attached to the EC2 instance grants permission to perform operations such as uploading, reading, and deleting objects in the S3 bucket.



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.2026.15348

The overall workflow of the architecture is as follows: a user accesses the web application through a browser. The request is processed by the application running inside the Docker container on the EC2 instance. When the application needs to store or retrieve data, it sends a request to the S3 bucket using the permissions provided by the IAM role. The S3 service then processes the request and returns the required data.

This architecture ensures secure interaction between the application and AWS services while maintaining scalability, flexibility, and efficient resource management.

VI. SYSTEM REQUIREMENTS

The successful implementation of the project Secure AWS Infrastructure Using Docker and IAM Roles requires several software tools and platforms that support application development, containerization, and interaction with AWS services. The primary platform used in this project is Amazon Web Services, which provides the infrastructure services required to host and manage the application. The application is deployed on a virtual server using Amazon EC2, which provides a flexible and scalable computing environment. Containerization of the application is achieved using Docker. Docker packages the application and its dependencies into a container, ensuring that the application runs consistently across different environments.

HARDWARE SPECIFICATIONS

Cloud Hardware (Virtual Instance):

- Instance Type: t2.micro / t3.micro (Free Tier Eligible)
- Processor: 1 vCPU (Virtual CPU)
- Memory (RAM): 1 GB minimum
- Storage: 8 GB – 20 GB Elastic Block Storage (EBS)
- Network: Virtual Private Cloud (VPC) enabled
- Internet Connectivity: Required for AWS service communication

Client System (for accessing AWS):

- Processor: Intel Core i3 or above
- RAM: Minimum 4 GB
- Storage: Minimum 20 GB free space
- Internet Connection: Required
- Device: Laptop/Desktop

SOFTWARE SPECIFICATIONS

Operating System:

- Amazon Linux 2023 (AL2023)

Cloud Platform:

- Amazon Web Services (AWS)

Cloud Services Used:

- EC2 (Elastic Compute Cloud) – Virtual server hosting Docker
- S3 (Simple Storage Service) – Secure storage service
- IAM (Identity and Access Management) – Role-based access control

Containerization Platform:

- Docker Engine

Programming / Command Tools:

- Bash Shell
- AWS CLI (Command Line Interface)
- Docker CLI

Web Browser:

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.2026.15348

- Google Chrome / Microsoft Edge

Security Mechanisms:

- IAM Roles for secure authentication
- Server-Side Encryption (SSE-S3)
- Docker Container Isolation
- Secure SSH access

Secure access control is implemented using AWS Identity and Access Management. IAM roles are used to grant permissions for accessing AWS services securely without storing access keys in the application code. The application interacts with AWS services using the AWS SDK for PHP, which provides libraries and APIs that allow the application to communicate with services such as S3. For application development, programming languages such as PHP, HTML, and CSS are used to create the backend functionality and user interface. Additionally, version control tools and package managers such as Composer are used to install and manage dependencies required by the AWS SDK and other libraries. These software components work together to provide a secure and efficient environment for developing, deploying, and managing the application infrastructure.

VII. IMPLEMENTATION

The implementation of the project Secure AWS Infrastructure Using Docker and IAM Roles involves setting up the required infrastructure, configuring secure access control, deploying the application inside a container, and enabling communication with AWS storage services.

1. AWS Infrastructure Setup
The first step in the implementation is setting up the infrastructure using services provided by Amazon Web Services. A virtual server is created using Amazon EC2 to host the application. The EC2 instance provides the computing environment where the application and Docker container will run. An object storage service is also configured using Amazon S3. A bucket is created in S3 to store and manage the application data. This bucket acts as the central storage location where the application uploads and retrieves files.
2. IAM Role Configuration
To maintain security, access to AWS services is controlled using AWS Identity and Access Management. An IAM role is created and assigned to the EC2 instance. The role contains policies that allow the application to perform specific operations such as reading and writing data to the S3 bucket. By using IAM roles, the application does not need to store permanent AWS credentials in its code. The permissions are automatically provided to the application through the role attached to the EC2 instance.

3. Docker Container Deployment
The application is containerized using Docker. A Dockerfile is created to define the container configuration, including the base image, application files, and required dependencies.

Example Dockerfile:

```
FROM php:8.2-apache
WORKDIR /var/www/html
COPY . /var/www/html
RUN apt-get update
RUN docker-php-ext-install mysqli
```

A Docker image is built from this Dockerfile, and a container is created to run the application. The container ensures that the application runs in an isolated environment with all necessary dependencies.

4. Application Integration with AWS Services
The application communicates with AWS services using the AWS SDK for PHP. The SDK provides APIs that allow the application to perform operations such as uploading, retrieving, and managing data in the S3 bucket. When the application needs to store data, it sends a request to the S3 bucket using the SDK. The request is authenticated using the IAM role attached to the EC2 instance.

5. Deployment and Testing

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.2026.15348

After configuring the infrastructure and container, the Docker container is started on the EC2 instance. The application becomes accessible through the EC2 public IP address. Various operations such as storing and retrieving data from S3 are tested to ensure that the IAM role permissions and container configuration work correctly. This implementation demonstrates how containerization, role-based access control, and AWS services can be integrated to build a secure and efficient infrastructure for application deployment.

VIII. RESULT AND DISCUSSION

The implementation of the project successfully demonstrates the use of secure infrastructure practices for application deployment. The application was deployed inside a Docker container running on an EC2 instance. The containerized environment ensured consistent execution and simplified application deployment.

Using IAM roles improved the security of the system by eliminating the need to store AWS access keys within the application code. The application was able to securely access the S3 bucket using permissions provided through IAM policies. The application successfully performed operations such as storing and retrieving data from the S3 bucket. The interaction between the application and AWS services was handled through the AWS SDK. The results show that combining Docker containerization with IAM-based access control provides a secure and reliable infrastructure for deploying applications.

ADVANTAGES OF THE SYSTEM

- Improved Security
  The project enhances infrastructure security by using role-based access control through AWS Identity and Access Management. IAM roles eliminate the need to store permanent credentials in the application code, reducing the risk of unauthorized access.

- Containerized Deployment
  The use of Docker allows the application and its dependencies to be packaged into a single container. This ensures that the application runs consistently across different environments and simplifies the deployment process.

- Scalable Infrastructure
  Hosting the application on Amazon EC2 provides a scalable computing environment. Resources such as CPU, memory, and storage can be adjusted based on application requirements.

- Reliable Data Storage
  The project uses Amazon S3 for storing application data. S3 offers high durability, reliability, and the ability to store large amounts of data efficiently.

- Simplified Application Management
  Containerization and cloud infrastructure simplify application management and maintenance. Updates and changes can be applied easily without affecting the entire system.

- Secure Integration with AWS Services
  The application communicates with AWS services using secure APIs through the AWS SDK. This ensures reliable and protected interaction between the application and AWS resources.

- Portability and Flexibility
  Because the application runs inside a Docker container, it can be easily moved or deployed to different environments without major configuration changes.

IX. CONCLUSION

The project Secure AWS Infrastructure Using Docker and IAM Roles demonstrates how a secure and efficient infrastructure can be implemented using services provided by Amazon Web Services. The application is deployed on a virtual server using Amazon EC2 and runs inside a containerized environment created with Docker. Containerization helps ensure consistent application performance and simplifies deployment across different environments.

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.2026.15348

Security is strengthened by implementing role-based access control through AWS Identity and Access Management. By using IAM roles, the application can securely access other AWS services without storing permanent credentials in the application code. This approach significantly reduces the risk of credential exposure and improves overall infrastructure security. The project also integrates Amazon S3 for storing and retrieving application data, providing reliable and scalable storage. Through the use of secure APIs and the AWS SDK, the application can communicate safely with AWS services.

Overall, this project successfully demonstrates how containerization and role-based access control can be combined to build a secure, scalable, and manageable infrastructure for deploying modern applications.

X. SCOPE FOR FUTURE ENHANCEMENT

Although the current system successfully demonstrates secure infrastructure deployment, several improvements can be made in the future to enhance the system further. One possible improvement is integrating monitoring and logging services such as AWS CloudWatch to monitor application performance and detect potential security issues. Another enhancement is implementing container orchestration tools such as Kubernetes to manage multiple containers and improve scalability.

Automated deployment pipelines can also be introduced using CI/CD tools to streamline the development and deployment process. Additional security measures such as network security groups, encryption mechanisms, and multi-factor authentication can also be implemented to further strengthen infrastructure security. These enhancements would make the system more robust, scalable, and suitable for large-scale production environments.

REFERENCES

[1]. Amazon Web Services Documentation – https://docs.aws.amazon.com
[2]. Docker Official Documentation – https://docs.docker.com
[3]. AWS Identity and Access Management User Guide – https://docs.aws.amazon.com/iam
[4]. Amazon EC2 Documentation – https://docs.aws.amazon.com/ec2
[5]. Amazon S3 Documentation – https://docs.aws.amazon.com/s3