International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Design and Deployment of a Highly Available Multi-Tier Web Application Using AWS Cloud Services

B. Hasini a, J. Ravinder a, K. Deepa a, K. V. Kyathieshwary a, Mrs. Jahanara Begum
Department of Science and Engineering, Hyderabad Institute of Technology and Management, Hyderabad, Telangana, India

Abstract—
Cloud-hosted applications can benefit from elastic infrastructure, managed data services, and centralized traffic management, but these capabilities depend on correct configuration and validation. This paper presents an implementation-oriented design for a multi-tier web application using Amazon Web Services (AWS). The architecture considers an Application Load Balancer, Amazon EC2, EC2 Auto Scaling, Amazon RDS, an Amazon Virtual Private Cloud, Security Groups, IAM, and CloudWatch. The methodology covers network setup, application hosting, traffic routing, database integration, access control, monitoring, and functional tests. An evidence-based test plan distinguishes resource configuration from observed behavior, particularly for scaling and availability. The paper provides a reproducible framework for evaluating application access, target health, database operations, scaling activity, and infrastructure metrics. Quantitative results and conclusions must be completed using measurements from the actual deployment.

Index Terms—AWS, cloud computing, EC2, multi-tier architecture, load balancing, Auto Scaling, Amazon RDS, CloudWatch.

I. INTRODUCTION

Cloud computing enables developers to provision compute, networking, storage, and database resources without operating all underlying physical infrastructure. Amazon Web Services (AWS) provides services that can be combined to host web applications and manage their operational requirements [1].

A single-server deployment can become a bottleneck as request volume grows, and the failure of that server may interrupt the service. Separating presentation, application processing, and data storage can improve modularity. When paired with suitable traffic distribution, capacity management, network controls, and monitoring, this structure provides a practical foundation for a more resilient deployment.

This paper describes an implementation-oriented approach to designing a multi-tier web application on AWS. The proposed design considers an Application Load Balancer (ALB), Amazon Elastic Compute Cloud (EC2), an EC2 Auto Scaling Group, Amazon Relational Database Service (RDS), an Amazon Virtual Private Cloud (VPC), AWS Identity and Access Management (IAM), Security Groups, and Amazon CloudWatch. Claims about availability, scaling, and performance are limited to the components deployed and the tests actually completed.

A. Objectives

The objectives are to (1) define a separated application and data architecture, (2) configure network boundaries, (3) deploy the application on EC2, (4) distribute requests through an ALB where implemented, (5) configure compute scaling where implemented, (6) connect a relational database, (7) apply least-privilege access controls, and (8) evaluate the environment using repeatable tests and monitoring evidence.

B. Scope and Contribution

The contribution is a practical deployment and evaluation framework that connects AWS service configuration to observable evidence. The study does not introduce a new cloud algorithm; instead, it records how architectural objectives can be checked through application responses, target health, database operations, scaling activity, and resource metrics.

II. RELATED WORK AND TECHNICAL BACKGROUND

AWS architecture guidance describes common patterns for hosting web applications by combining networking, compute, data, and operational services [1]. The specific design depends on workload, availability targets, security needs, and cost constraints.

Elastic Load Balancing routes requests to registered targets and uses configured health checks to report target health [2]. EC2 Auto Scaling manages groups of instances within defined capacity limits and can adjust capacity when configured policies are triggered [3]. These services can complement one another, but their presence alone does not prove that requests are balanced effectively or that a scaling event has occurred.

Amazon VPC provides virtual networking constructs, including subnets, route tables, and connectivity controls [4]. Security Groups provide stateful traffic filtering, while IAM policies govern permitted AWS actions. CloudWatch collects supported metrics and can support alarms and operational investigation [5]. Amazon RDS provides managed relational database capabilities, but access, backup, and availability settings still require deliberate configuration [6].

The practical gap addressed here is not a lack of service documentation; it is the need to validate the integrated deployment. The evaluation therefore distinguishes configuration evidence from observed behaviour and reports untested features as such.

III. SYSTEM ARCHITECTURE

The system is organized into a presentation interface, an application tier, and a data tier. A user sends an HTTP or HTTPS request to the configured endpoint. If an ALB is deployed, it evaluates listener rules and forwards requests to eligible targets in a target group. EC2 instances run the application logic and connect to RDS for database operations. The resources are placed within a VPC and governed by network rules and IAM permissions. CloudWatch supports monitoring of available service metrics.

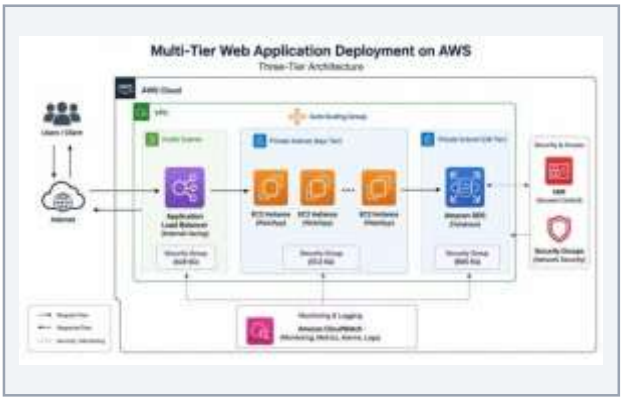


FIGURE 1. Logical architecture: User → ALB (if deployed) → Target Group → EC2 application instances (optionally managed by Auto Scaling) → Amazon RDS. VPC, subnets, route tables, Security Groups, IAM, and CloudWatch provide networking, access control, and monitoring. The final diagram must include only components verified in the deployed environment.

A. Presentation and Application Tiers

Configure the database engine and deployment options, subnet group, Security Group, and authentication method. Supply connection information securely to the application. From the The presentation tier is the browser-facing interface. The application tier processes requests and returns responses. Each EC2 instance must contain the required runtime, web server, application code, and configuration. When multiple instances are used, their application versions and required dependencies should be consistent.

B. Traffic Management and Scaling

An ALB can provide a stable entry point and forward traffic to registered targets according to listener and target-group configuration [2]. An Auto Scaling Group can maintain desired capacity within minimum and maximum limits [3]. The deployment should record target health, capacity settings, scaling policies, and activity history. High availability requires more than an ALB: instance placement, application dependencies, database configuration, and failure recovery also matter.

C. Data Tier

RDS stores structured application data and exposes a database endpoint to permitted clients [6]. The database should be placed in the intended subnets, and its Security Group should allow connections only from the application tier where practical. Successful integration is demonstrated by a connection and a query or application operation—not merely by an RDS status of "available."

D. Network and Security

The VPC contains selected network resources. Public-facing components require appropriate routes and connectivity, while application and database resources can be placed in more restricted subnets according to the design. Security Group rules should permit only necessary traffic. IAM permissions should follow the least privilege. Credentials and private keys must not be included in public screenshots, source repositories, or the paper.

IV. IMPLEMENTATION METHODOLOGY

Implementation proceeds through requirement definition, network configuration, compute deployment, traffic configuration, database integration, security review, monitoring, and functional validation. Resource identifiers, configuration values, and test evidence should be collected from the actual AWS environment.

A. Network Setup

Create or select a VPC, configure subnets and route tables, and associate required gateways. Confirm that each resource is attached to the intended subnet and Security Group. Record the AWS Region, CIDR ranges, Availability Zones, and routing choices. Do not publish sensitive account information.

B. EC2 and Application Deployment

Select an appropriate machine image and instance type, configure access, install the required application at runtime, and deploy the application. Verify the service is listening on its intended port, and that permitted network traffic can reach it. Record the operating system, instance type, software versions, and deployment steps.

C. ALB and Target Group

If implemented, configure the ALB scheme, listeners, target group, health-check path, protocol, and port. Register the application instances and inspect target health. Validate access through the ALB endpoint and retain evidence of the response and target status. If this component was not deployed, report the test as not applicable rather than completed.

D. Auto Scaling

If implemented, configure a launch template, subnet placement, minimum/desired/maximum capacity, health-check settings, and a scaling policy. Use the Auto Scaling Activity history to verify a scaling event. If no event was induced or observed, report only the configuration review and do not claim automatic scaling was demonstrated.

E. RDS Integration

application environment, establish a connection and execute a representative query or database-backed operation. Record the operation and a redacted summary of its output.

F. Monitoring

Use CloudWatch to inspect relevant metrics during a stated time window. Depending on deployed resources, these may include EC2 CPU utilization, load balancer request or target metrics, and RDS resource utilization. Record metric names, timestamps, workload conditions, and alarm states if alarms exist. A graph without a defined workload is an observation, not a controlled benchmark.



Figure 2. Application Load Balancer and target health screenshot.

V. TESTING AND EVALUATION

Testing should be performed against the deployed configuration and recorded with an expected outcome, observed result, and status. A test is marked Pass only when the expected outcome is observed; otherwise it is Fail or Not Tested. The environment and test period should be reported to make the results interpretable.

A. Test Cases

Table 1 – FUNCTIONAL TEST CASES

ID	Test	Expected evidence	Status
T01	Application access	Expected page/response	[Pass]
T02	Target health	Expected targets healthy	[Pass]
T03	Traffic distribution	Requests reach targets	[Pass]
T04	Auto Scaling	Capacity change and activity record	[Pass]
T05	Instance replacement	Replacement evidence when configured	[Pass]
T06	Database connection	Connection from application tier	[Pass]
T07	Database operation	Expected query/output	[Pass]
T08	Network access	Rules match intended access	[Pass]
T09	CloudWatch metrics	Metrics visible for stated period	[Pass]
T10	Performance	Measured values under defined load	[Pass]

Traffic distribution should be verified separately from target health, for example through request logs or instance-specific responses. Performance testing requires a defined workload, tool, duration, concurrency, and measurements such as response time and error rate. If those conditions were not established, performance remains unevaluated.



Figure 3. CloudSphere dashboard screenshot.

VI. RESULTS AND DISCUSSION

The CloudSphere project demonstrates an implementation-oriented approach to deploying a multi-tier web application using Amazon Web Services (AWS). The proposed architecture integrates application hosting, traffic management, database connectivity, network security, and infrastructure monitoring. The deployment evaluation focuses on application accessibility, load balancer configuration, target health, database connectivity, and monitoring activities. The results are assessed using the configuration details and testing evidence collected from the AWS environment.

A. Functional Results

The functional evaluation examines whether the deployed application can be accessed through its configured endpoint. The Application Load Balancer and target group configurations are reviewed to verify the health status of registered application instances. Database connectivity is evaluated by checking whether the application environment can establish a connection with the configured relational database. The final test outcomes are classified as Pass, Fail, Not Tested, or Not Applicable according to the observed results.



Figure 4. Functional testing and AWS monitoring evidence.

B. Availability and Scaling Interpretation

The Cloud Sphere architecture uses an Application Load Balancer and EC2-based application hosting to support traffic distribution. EC2 Auto Scaling can provide automatic capacity adjustment when the appropriate scaling policies are configured. The effectiveness of availability and scaling depends on the actual deployment configuration, healthy target status, instance placement, and observed scaling activities. Therefore, high availability is evaluated based on deployment evidence rather than service configuration alone.

C. Monitoring and Performance

Amazon CloudWatch is considered for monitoring the AWS infrastructure. Relevant monitoring observations include EC2 CPU utilization, load balancer metrics, target health information, and RDS resource utilization, depending on the services deployed. Performance evaluation requires measurements collected under a defined workload. Response time, throughput, and error rate should be recorded when controlled performance testing is conducted.

D. Evidence-Based summary

project evaluation connects the objectives of the Cloud Sphere architecture with the results obtained from functional testing and AWS monitoring. Each objective is classified according to its implementation status and available evidence.The evaluation distinguishes between configured AWS services and successfully demonstrated application behaviour. Any untested functionality or incomplete deployment activity is identified as a limitation.

VII. LIMITATIONS AND FUTURE WORK

This evaluation is bounded by the deployed configuration, available resources, and duration of testing. Short functional checks do not establish long-term availability, performance at scale, backup recovery, or disaster recovery. Load balancing, scaling, and managed database services require correct configuration and must be assessed in the context of application dependencies.

Future work may include controlled load testing, multi-Availability Zone failure exercises, database backup restoration, and recovery-time measurement. Infrastructure as code using AWS CloudFormation or Terraform could improve repeatability. A CI/CD pipeline could automate build and deployment checks. Additional controls such as AWS WAF or content delivery through CloudFront may be considered when justified by the application's requirements.

VIII. CONCLUSION

This paper presented a practical approach for designing and evaluating a multi-tier web application deployment using AWS services. The architecture separates application

processing from relational data storage and considers network controls, request distribution, compute capacity management, and monitoring. The methodology defines how to configure and validate the environment, while the test plan separates application access, target health, database operations, scaling behaviour, and metrics. Final conclusions must be based on the project's actual configuration and observed test results. Controlled workload tests, failure experiments, and recovery validation are necessary before making broader claims about performance, high availability, or resilience.

REFERENCES

[1] Amazon Web Services, "Web Application Hosting in the AWS Cloud," AWS Whitepapers. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/web-application-hosting-best-practices/welcome.html

[2] Amazon Web Services, "What Is an Application Load Balancer?" Elastic Load Balancing User Guide. [Online]. Available: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

[3] Amazon Web Services, "What Is Amazon EC2 Auto Scaling?" Amazon EC2 Auto Scaling User Guide. [Online]. Available: https://docs.aws.amazon.com/autoscaling/ec2/userguide/

[4] Amazon Web Services, "What Is Amazon VPC?" Amazon Virtual Private Cloud User Guide. [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/

[5] Amazon Web Services, "What Is Amazon CloudWatch?" Amazon CloudWatch User Guide. [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

[6] Amazon Web Services, "What Is Amazon RDS?" Amazon Relational Database Service User Guide. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html