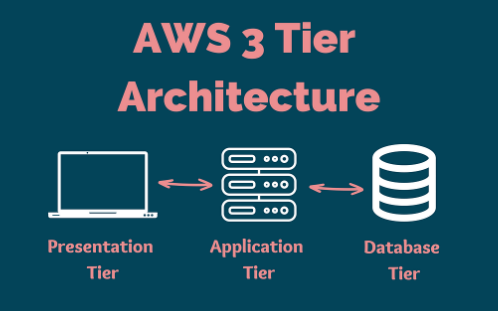
AWS 3-Tier Architecture Project

# Table of Contents

1. Introduction
2. Brief Idea About the Project
3. Benefits of a Three-Tier Architecture
4. AWS Services and Resources Used
5. Implementation Process
   * 5.1 Create a VPC
   * 5.2 Set Up Subnets
   * 5.3 Configure Internet Gateway and NAT Gateway
   * 5.4 Create Route Tables and Associate with Subnets
   * 5.5 Launch Template and Auto Scaling Groups
   * 5.6 Application Load Balancer Setup
   * 5.7 Amazon RDS for Database Tier
   * 5.8 Testing the Architecture
6. Best Practices
7. Conclusion

## 1. Introduction

In this project, we implement a **three-tier architecture** on Amazon Web Services (AWS), which provides a scalable, secure, and resilient cloud infrastructure for a web application. The project utilizes several AWS services to design a highly available system distributed across multiple availability zones. Each tier in the architecture is responsible for a different functionality:

****

* **Web Tier**: Accepts user traffic and forwards it to the application.
* **Application Tier**: Hosts the business logic of the application.
* **Database Tier**: Stores and manages application data.

This architecture ensures high availability, fault tolerance, and scalability by leveraging AWS services like EC2, Auto Scaling Groups, Application Load Balancer (ALB), and Amazon RDS.

## 2. Brief Idea About the Project

The goal of the project is to build a secure, scalable, and highly available **three-tier web application** architecture on AWS. The architecture is composed of:

* A **web tier** hosted in EC2 instances, distributed across public subnets in two availability zones.
* An **application tier** hosted in private EC2 instances, only accessible by the web tier.
* A **database tier** using **Amazon RDS** for persistent data storage in private subnets.

The architecture is built with fault tolerance in mind, using **Auto Scaling Groups** to automatically adjust the number of EC2 instances based on traffic and **Load Balancers** to distribute traffic across instances.

## Architecture diagram –

A diagram of a network

Description automatically generated

## 3. Benefits of a Three-Tier Architecture

Implementing a three-tier architecture on AWS offers several advantages:

* **Scalability**: Auto Scaling Groups automatically adjust the number of EC2 instances based on incoming traffic, ensuring efficient resource usage.
* **High Availability**: By deploying resources across multiple Availability Zones, the architecture achieves high availability and resilience against outages.
* **Security**: Each tier is isolated with dedicated security groups, ensuring that sensitive resources like databases remain protected from public access.
* **Fault Tolerance**: In case of instance or zone failures, the architecture ensures that other instances in different AZs handle the load.

## 4. AWS Services and Resources Used

The architecture is built using several AWS services, each contributing to different layers of the application:

* **Elastic Compute Cloud (EC2)**: Virtual servers hosting the application in the web and application tiers.
* **Auto Scaling Groups (ASG)**: Automatically scales EC2 instances based on traffic demand.
* **Virtual Private Cloud (VPC)**: Provides isolated networking for your AWS resources.
* **Subnets**: Divides the VPC into multiple subnets (public and private) across Availability Zones.
* **Application Load Balancer (ALB)**: Distributes incoming traffic across multiple EC2 instances in the web and application tiers.
* **Internet Gateway and NAT Gateway**: Allows access to the internet for public subnets and routes outbound traffic from private subnets.
* **Amazon Relational Database Service (RDS)**: Manages and provides secure and scalable relational databases.
* **Security Groups**: Acts as a firewall to control inbound and outbound traffic for each tier.

## 5. Implementation Process

**5.1 Create a VPC**

A **Virtual Private Cloud (VPC)** is the core networking component of AWS. It provides isolated network space for your AWS resources.

* **VPC CIDR**: 10.0.0.0/16 (providing ample IP address space).
* A **VPC** is created to isolate and control the network configuration, ensuring better management and security of the application.

**5.2 Set Up Subnets**

Subnets divide the VPC into smaller networks that can be allocated for specific purposes.

* **Public Subnets**:
  + Subnet 1: 10.0.0.0/20 (Availability Zone: us-east-1a).
  + Subnet 2: 10.0.16.0/20 (Availability Zone: us-east-1b).
* **Private Subnets**:
  + Subnet 3: 10.0.128.0/20 (App tier).
  + Subnet 4: 10.0.144.0/20 (App tier).
  + Subnet 5: 10.0.160.0/20 (DB tier).
  + Subnet 6: 10.0.176.0/20 (DB tier).

**5.3 Configure Internet Gateway and NAT Gateway**

* **Internet Gateway** is created and attached to the VPC to allow public subnets to access the internet.
* **NAT Gateway** is set up in one of the public subnets to allow EC2 instances in private subnets to access the internet for outbound traffic.

**5.4 Create Route Tables and Associate with Subnets**

* **Public Route Table**: Configured to route all internet-bound traffic through the Internet Gateway.
* **Private Route Table**: Routes internet-bound traffic from private subnets via the NAT Gateway.

**5.5 Launch Template and Auto Scaling Groups**

* **Launch Template**: Defines the EC2 configuration, including the AMI, instance type, security group, and a bootstrap script to install web server software (e.g., Apache or Nginx).
* **Auto Scaling Groups (ASG)**:
  + Web Tier: Automatically scales EC2 instances across public subnets.
  + Application Tier: EC2 instances scale based on demand in private subnets.

**5.6 Application Load Balancer Setup**

* An **Application Load Balancer (ALB)** is deployed in public subnets. It balances incoming HTTP requests across the EC2 instances in the web tier.
* The ALB uses health checks to monitor the state of instances and routes traffic only to healthy instances.

**5.7 Amazon RDS for Database Tier**

* **Amazon RDS** is used to create a MySQL database hosted in private subnets. The RDS instance is not accessible from the public internet, providing a secure database environment.
* Multi-AZ deployment is enabled for RDS to ensure high availability and failover support.

**5.8 Testing the Architecture**

1. **Web Access**: Validate the application by accessing the ALB's DNS name.
2. **Auto Scaling**: Simulate load and monitor the auto-scaling behavior.
3. **Database Connectivity**: Ensure the web application can connect to the RDS instance for data storage and retrieval.

## 6. Best Practices

* **Use of Multi-AZ**: Deploying resources (EC2, RDS) in multiple availability zones ensures high availability and fault tolerance.
* **Security Groups**: Each tier should have its own security group to restrict traffic between the web, application, and database tiers.
* **Encryption**: Enable encryption at rest for databases and use SSL/TLS for secure communication between tiers.
* **Monitoring**: Utilize CloudWatch to monitor performance metrics and set up alarms for critical issues.
* **Cost Optimization**: Utilize AWS Free Tier for testing, and right-size instances based on traffic demands.

## 7. Conclusion

The three-tier architecture designed on AWS offers a scalable, resilient, and secure infrastructure for running production-level applications. By leveraging AWS services such as **EC2**, **RDS**, **ALB**, and **Auto Scaling**, the architecture ensures high availability, fault tolerance, and cost-efficiency. This well-structured deployment allows each tier to function independently, providing the flexibility to scale resources up or down based on usage patterns.

The use of best practices such as multi-AZ deployments, security groups, and encryption ensures that the architecture is not only scalable but also secure and reliable for handling traffic at any scale.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*