A Mini Project Report

on

"AWS 3-Tier Architecture with Public and Private Subnets and Load Balancers"

Submitted to

CLOUD COMPUTING LAB
(20BT61201)

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Submitted by

G Y SAI MANOJ	**21121A0561**
GADDAM MANJUNATH REDDY	**21121A0562**
GADIPARTHI PRABHAS CHOWDARY	**21121A0563**
GAJULA MOHANTEJA ROYAL	**21121A0564**
GOVINDA PRASAD YADAV	**21121A0565**
JYOTI KUMARI CHAUDHARY	**21121A0566**
SANDIP MALI	**21121A0567**
SMRITEE POUDEL	**21121A0569**

GANDLA NAGA SUBRAMANYAM	21121A0570
GANGULA CHAITANYA	21121A0571
GANIMANENI SAI JASMITHA	21121A0572
GODULA GAYATHRI	21121A0573
GOLLA PRADEEP KUMAR	21121A0574
GOLLA RAJESH	21121A0575

Department of Information Technology

SREE VIDYANIKETHAN ENGINEERING COLLEGE

(AUTONOMOUS)

(Affiliated to JNTUA, Ananthapuramu, Approved by AICTE, Accredited by NBA
& NAAC) Sree Sainath Nagar, Tirupati – 517 102, A.P., INDIA

2023-2024

AWS 3-Tier Architecture with Public and Private Subnets and Load Balancers

ABSTRACT

In modern cloud computing, deploying web applications within a secure, scalable, and robust architecture has become paramount. This project explores a 3-tier architecture implemented in Amazon Web Services (AWS), emphasizing the benefits of a hybrid public-private VPC model with private EC2 instances. By utilizing an Application Load Balancer (ALB) to interface with private subnets, this architecture prioritizes security while achieving efficient load distribution and resilience. The setup encompasses distinct layers: a web tier, an application tier, and a database tier, each isolated within private subnets and accessed via the load balancer, which mitigates direct public exposure. Key elements such as NAT Gateways, Security Groups, and VPC configurations are explored to ensure a controlled, internet-facing entry point, balancing accessibility with data integrity. Challenges such as secure network routing and fault tolerance are addressed by leveraging AWS services to manage dynamic scaling and workload distribution. This project ultimately demonstrates an effective approach to deploying web applications securely, illustrating best practices in designing scalable and cost-efficient cloud infrastructures.

Keywords: AWS, VPC, EC2, Application Load Balancer, NAT Gateway, Security Groups, 3-Tier Architecture.

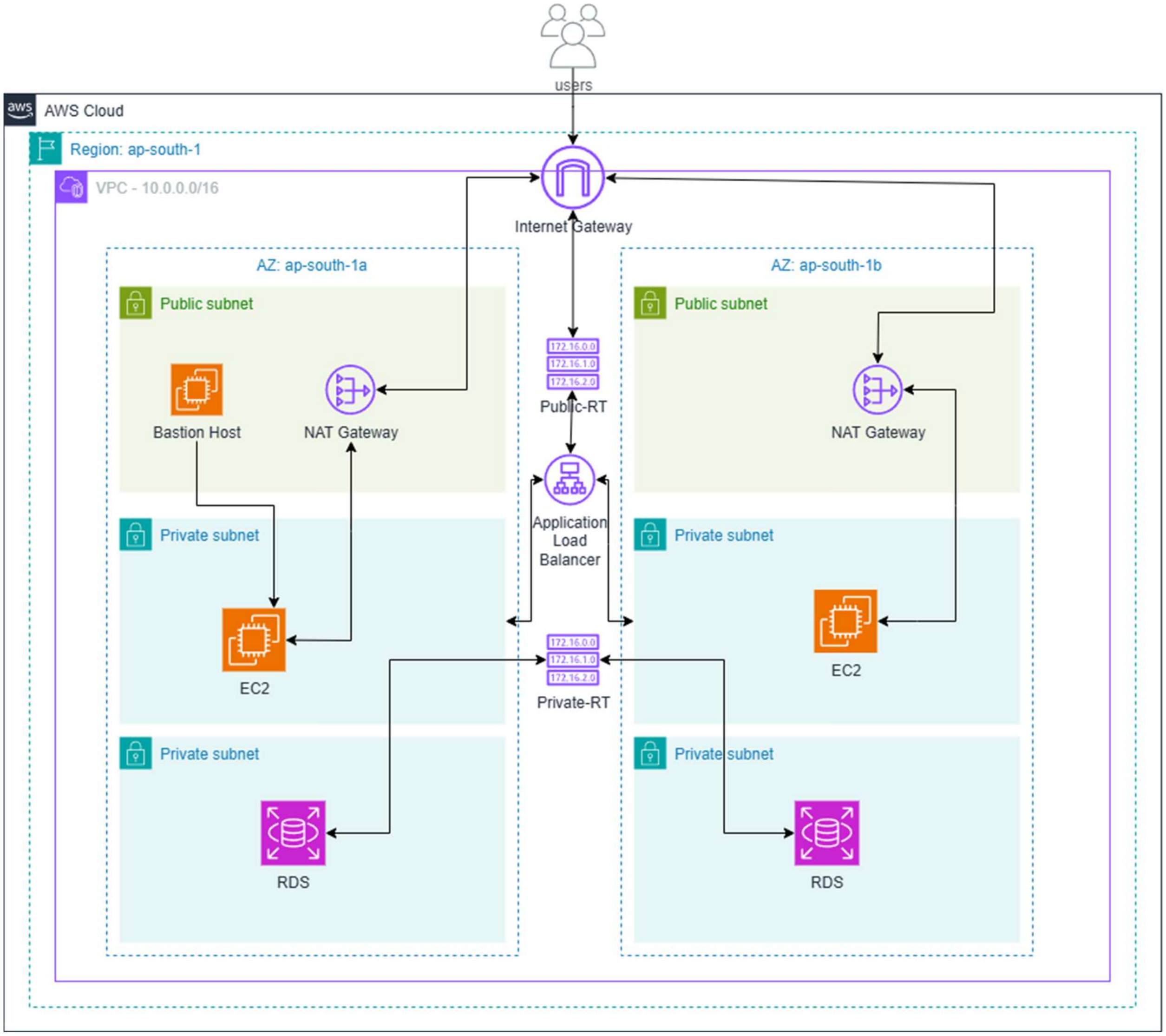
INTRODUCTION

Cloud computing has transformed the landscape of application deployment, offering robust frameworks that enhance scalability, security, and cost-effectiveness. Among these, Amazon Web Services (AWS) provides a versatile environment for implementing multi-tier architectures, a preferred model for complex applications requiring clear separation between presentation, application processing, and data management layers. In this project, we deploy a secure 3-tier architecture within AWS, leveraging a Virtual Private Cloud (VPC) structure that segregates the web, application, and database layers into private subnets, thereby reducing direct exposure to the internet and bolstering overall system security.

The architecture utilizes an Application Load Balancer (ALB) as the single access point, directing traffic to private EC2 instances distributed across the web and application tiers. By offloading internet access management to the ALB, the architecture achieves both security and seamless load distribution, accommodating fluctuating traffic loads without compromising performance. Furthermore, essential AWS services, such as NAT Gateways for internet-bound access and Security Groups for controlled inbound and outbound rules, are implemented to facilitate internal communications while restricting unauthorized access.

This project aims to illustrate how a well-designed 3-tier architecture can enhance the resilience, security, and scalability of cloud-deployed applications. By following industry best practices for network isolation, load balancing, and controlled public access, this setup showcases an optimized approach to building cloud-native web applications suitable for diverse use cases, including e-commerce, finance, and enterprise software solutions.

RESOURCE DIAGRAM



RESOURCES LIST

1. VPC
2. Subnets
3. Internet Gateway
4. NAT Gateway
5. Route Tables
6. Application Load Balancer (ALB)
7. EC2 Instances
8. Amazon RDS
9. Security Groups

Procedure:

1. Create a Virtual Private Cloud (VPC)

1. **Go to VPC Console**:
    - Navigate to **Services** > **VPC** > **Your VPCs**.
2. **Create a New VPC**:
    - Click **Create VPC**.
    - **Name**: My-Private-VPC.
    - **IPv4 CIDR Block**: 10.0.0.0/16 (or an appropriate range for your network).
    - **Tenancy**: Set to **Default** (unless you require dedicated instances).
    - Click **Create VPC**.

2. Create Subnets (Public for NAT Gateway and Private for Instances)

1. **Navigate to Subnets**:
   - Click **Subnets** on the left sidebar.
2. **Create Public Subnet** (for NAT Gateway only):
   - Click **Create Subnet**.
   - **Name**: Public-Subnet-NAT.
   - **VPC**: Choose My-Private-VPC.
   - **Availability Zone (AZ)**: Select an AZ (e.g., us-east-1a).
   - **IPv4 CIDR Block**: Use 10.0.1.0/24.
   - Click **Create Subnet**.
3. **Create Private Subnets** (for EC2 Instances):
   - Repeat steps to create private subnets for each tier:
     - **Public-Subnet-Web** in 10.0.2.0/24.
     - **Private-Subnet-App** in 10.0.3.0/24.
     - **Private-Subnet-DB** in 10.0.4.0/24.
   - Assign different AZs to achieve high availability, e.g., us-east-1b and us-east-1c.

3. Create an Internet Gateway (IGW)

1. **Navigate to Internet Gateways**:
   - In the VPC console, click on **Internet Gateways**.
2. **Create IGW**:
   - Click **Create Internet Gateway**.

  - **Name**: My-IGW.
  - Click **Create Internet Gateway**.

3. **Attach IGW to VPC**:
   - Select your IGW, click **Actions** > **Attach to VPC**.
   - Choose My-Private-VPC and click **Attach**.

4. Configure Route Tables

a. Create a Route Table for Public Subnet (for NAT Gateway)

1. **Create Public Route Table**:
   - Go to **Route Tables** > **Create Route Table**.
   - **Name**: Public-RT.
   - **VPC**: Choose My-Private-VPC.
   - Click **Create**.
2. **Add Route for Internet Access**:
   - Select Public-RT, go to the **Routes** tab, and click **Edit routes**.
   - **Destination**: 0.0.0.0/0
   - **Target**: Select **Internet Gateway** and choose My-IGW.
   - Click **Save changes**.
3. **Associate Route Table with Public Subnet**:
   - Go to **Subnet Associations** tab, click **Edit subnet associations**.
   - Select **Public-Subnet-NAT** and click **Save associations**.

b. Create a Route Table for Private Subnets

1. **Create Private Route Table**:
    - Repeat the above steps to create a new route table called Private-RT.
    - No additional routes needed yet; only assign private subnets in **Subnet Associations**.

5. Create NAT Gateway

1. **Go to NAT Gateways**:
    - Click on **NAT Gateways**.
2. **Create NAT Gateway**:
    - **Subnet**: Select Public-Subnet-NAT.
    - **Elastic IP**: Allocate a new Elastic IP and assign it.
    - Click **Create NAT Gateway**.
3. **Edit Routes in Private Route Table**:
    - Go to **Route Tables** > Select Private-RT > **Edit Routes**.
    - **Destination**: 0.0.0.0/0
    - **Target**: Select the **NAT Gateway** created.
    - Click **Save changes**.

6. Security Groups Setup

1. **Create Security Groups for Each Tier**:
    - **Bastion SG**: SSH (Port 22) from your IP.
    - **Web Tier SG**: HTTP (80) and HTTPS (443) from the ALB security group.
    - **App Tier SG**: Port 8080 (or custom) from Web Tier SG.

  - **DB Tier SG**: Database port (e.g., 3306 for MySQL) from App Tier SG.

7. Create Private EC2 Instances

1. **Launch EC2 Instances in Private Subnets**:
   - Create instances in the **Private-Subnet-Web** for the Web Tier, **Private-Subnet-App** for Application Tier, and **Private-Subnet-DB** for Database Tier.
   - For each instance:
     - Choose an AMI (e.g., Amazon Linux).
     - **Subnet**: Select the appropriate private subnet.
     - **Security Group**: Attach relevant SG (e.g., Web-SG for Web Tier).

8. Create a Bastion Host

1. **Launch Bastion Host in Public Subnet**:
   - Launch an EC2 instance in **Public-Subnet-NAT**.
   - **Name**: Bastion-Host.
   - Attach **Bastion-SG** to it, allowing SSH from your IP.
2. **Connect to Private EC2 Instances**:
   - Use the Bastion Host to SSH into private instances by SSHing from the Bastion to private IPs of the instances.

9. Create an Application Load Balancer (ALB)

1. **Navigate to Load Balancers**:
   - Go to **EC2 Console** > **Load Balancers** > **Create Load Balancer**.
   - Choose **Application Load Balancer**.

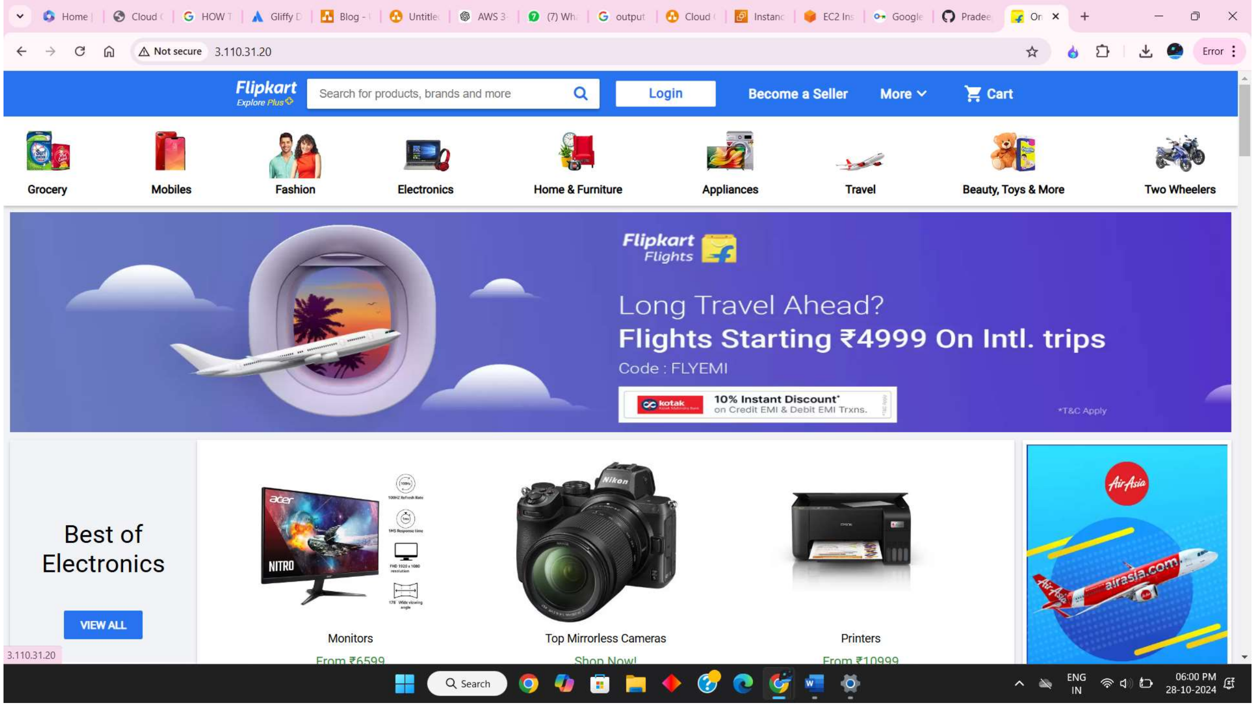
2. **Configure the ALB**:
   - **Name**: App-ALB.
   - **Scheme**: **Internet-facing**.
   - **Availability Zones**: Choose **private subnets** only.
   - **Security Group**: Assign Web-Tier-SG.
3. **Set Up Listeners**:
   - Create listeners for HTTP (80) and/or HTTPS (443).
4. **Configure Target Group**:
   - **Create Target Group**: Register **Web Tier** private EC2 instances in the target group.
   - Ensure health checks are configured properly (e.g., /health endpoint).

10. Test the Setup

1. **Access the ALB**:
   - Copy the **DNS name** of the ALB and try accessing it from your browser.
   - Ensure traffic routes to the private EC2 instances without public exposure.
2. **Verify Security and Traffic Flow**:
   - Confirm that private EC2 instances are accessible only via the ALB.
   - Check that security groups allow only necessary traffic.

Output:

Successfully implemented three tier architecture and deployed a web application for testing.



Conclusion:

Implementing a 3-tier architecture on AWS provides a highly scalable, secure, and efficient solution for deploying web applications. By segregating the web, application, and database layers into separate subnets and isolating them within a Virtual Private Cloud (VPC), this architecture ensures robust security while enabling efficient load distribution via an Application Load Balancer (ALB). The use of private EC2 instances for both the application and database tiers, with access managed through NAT Gateway and Security Groups, strengthens security and compliance by restricting public exposure. This architecture offers a flexible framework suitable for various applications, including enterprise solutions, e-commerce platforms, and more, showcasing best practices in cloud infrastructure design for high availability, fault tolerance, and cost-efficiency.

References:

1. https://aws.amazon.com/architecture/
2. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Subnets.html
3. https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
4. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html
5. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html
6. https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html