Multi-Tier Website using AWS EC2 Instance

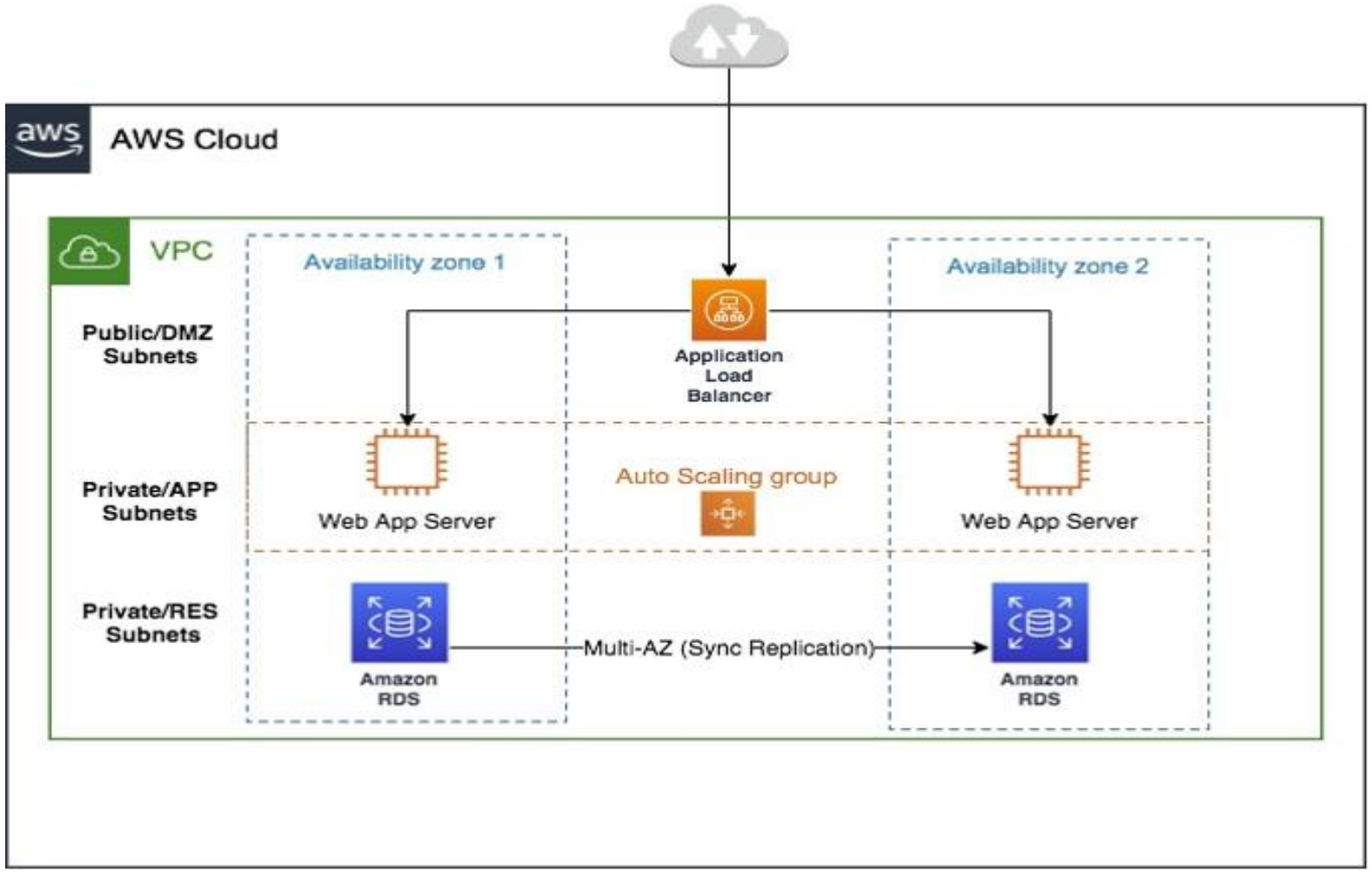
Project Overview:-

- This project involves deploying a Multi-Tier Website using AWS EC2 Instance.
- Using Amazon EC2 eliminates your need to invest in hardware up front so you can develop and deploy applications faster.
- The multi-tier website project leverages AWS EC2 instances to establish a robust architecture encompassing presentation, application.

Services Used:-

AWS EC2, VPC, RDS Instance, Route 53, ELB, Autoscaling, CloudWatch.

Architecture diagram :-



Explanation:-

Step 1: Sign in to AWS Console

Log in to the AWS Management Console at https://aws.amazon.com/.

Step 2: Launch an EC2 Instance

- Navigate to the EC2 Dashboard.
- Click "Launch Instance" to create a new instance.
- Choose an Amazon Machine Image (AMI) based on your requirements.
- Select an instance type.
- Configure instance details, including the number of instances, network settings, and storage.
- Add tags (optional but recommended).
- Configure security groups to allow inbound traffic for HTTP, HTTPS, and SSH.
- Review and launch the instance.

Amazon VPC Setup:

- Configuration details of the Virtual Private Cloud.
- Subnets(Public and private subnet), route tables, and security groups.
- create the public and private subnets for the instance.

Step 3: Connect to EC2 Instance

- User can connect to the ec2 instance with the help of PUTTY (SSH client)

(OR)

- Once the instance is running, select it in the EC2 Dashboard.
  Click "Connect" to get connection instructions.
  Use SSH to connect to your instance. (In AWS Management console)

Step 4: Install Web Server (e.g., Apache)

- Install a web-server on EC2 instance through the CLI.
- web-server might be an Apache or Nginx.

Step 5: Set up RDS Instance

- Navigate to the RDS Dashboard.
- Click "Create database" and select the engine (e.g., MySQL, PostgreSQL).
- Configure DB instance details, including DB instance identifier, master username, and password.
- Set up advanced settings, including VPC, subnet group, and security group.
- Review and launch the RDS instance.

Step 6: Connect Web Server to RDS

- By using the RDS endpoint we can connect to the EC2 instance.
- Update your web application configuration to use the RDS endpoint, database name, username, and password.

Step 7: Test Your Setup

- Visit your EC2 instance's public IP or domain name in a web browser to test if your multi-tier website is working.

Step 8: Set Up Auto Scaling and Load Balancing

- If you want to improve reliability and handle varying loads, consider setting up auto scaling and a load balancer.

Step 9: Configure Domain Name

- Register a domain name (e.g., using Route 53).
- Associate the domain name with your EC2 instance or load balancer.

Step 10: Monitor and Scale

- Regularly monitor your instances using AWS CloudWatch. Adjust resources, scale your instances, and optimize based on traffic patterns.

Step 11: Test the setup

- Test the entire setup once …, so that no error can occur when the traffic enters into the environment.

Step 12 : AWS IAM(Identity and Access Management)

- Use the AWS IAM service so that .. The organization/company can control the which resources can used by the end-users.

Challenges Faced during Implementation:

1. Latency and Response Time.
2. Content Delivery.
3. Security Concerns.

⚙ Solutions For Faced Challenges:-

Latency and Response Time:

- As requests pass through multiple tiers, latency can be introduced, affecting the overall response time of the website.
- Optimizing communication and minimizing round-trip delays are essential.

Content Delivery:

- Efficient content delivery, including static assets like images, stylesheets, and scripts, is crucial for performance. Utilizing content delivery networks (CDNs) .
- optimizing asset loading can be challenging but is necessary for a smooth user experience.

Future considerations 🔮 :-

- The future considerations for multi-tier websites will likely revolve around emerging technologies,evolving user expectations, and industry trends.

Here are some key aspects to consider,

1. Serverless Computing.
2. AI and Machine Learning Integration.
3. Decentralized Web and Blockchain.

Conclusion:-

🔑 Future-proofing multi-tier websites demands strategic adoption of microservices, serverless computing, and AI integration.

- The multi-tier development ensures resilience, scalability, and user-centricity in an evolving technological landscape.