2-Tier Web Architecture Deployment on AWS

**Project Title:**

Deploying a Portfolio Website on a Scalable 2-Tier Architecture Using Apache Tomcat and Load Balancer.

**Project Overview**

This project demonstrates the deployment of a scalable 2-tier web architecture on AWS. The application layer comprises Apache Tomcat servers serving static HTML-based portfolio websites, and traffic distribution is managed using an AWS Load Balancer. This setup simulates a real-world scenario for scalable web hosting.

**Technologies Used**

* Cloud Provider: AWS
* Compute: EC2 Instances (Amazon Linux 2)
* Web Server: Apache Tomcat
* Load Balancer: AWS Application Load Balancer (ALB)
* Language/Content: HTML/CSS Portfolio Projects

**Implementation Steps**

**1. Launch EC2 Instances**

• Created 3 EC2 instances in the same VPC and availability zone.  
• Selected Amazon Linux 2 as the OS.  
• Configured security group to allow:  
 - Port 22 (SSH)  
 - Port 80 (HTTP)

**2. Install Apache Tomcat**

Connected to each instance via SSH and executed:

sudo su

yum install http -y

**3. Deploy Portfolio Code**

• Replaced the default index.html file in the webapps/ROOT directory on each instance.  
• Each instance contains a different version of the portfolio (e.g., Portfolio A, B, and C).

**4. Set Up Load Balancer**

• Created an Application Load Balancer in the same region and VPC.  
• Added all three EC2 instances to the target group.  
• Configured a listener on port 80 to forward traffic to the target group.  
• Verified health checks (target group health checks passed).

**Testing**

• Accessed the Load Balancer's public DNS name.  
• On refreshing, different versions of the portfolio appeared due to round-robin routing.  
• This confirmed successful load balancing and server distribution.

**Outcome**

• Successfully deployed a scalable 2-tier architecture.  
• Demonstrated:  
 - Load balancing of web traffic.  
 - Hosting multiple versions of a website.  
 - Apache Tomcat configuration and custom deployment.

**Future Improvements**

* Integrate with RDS for dynamic content (transition to 3-tier architecture).
* Add Auto Scaling Groups for high availability.
* Use CI/CD pipeline to automate portfolio updates.
* Implement HTTPS with SSL/TLS certificates.