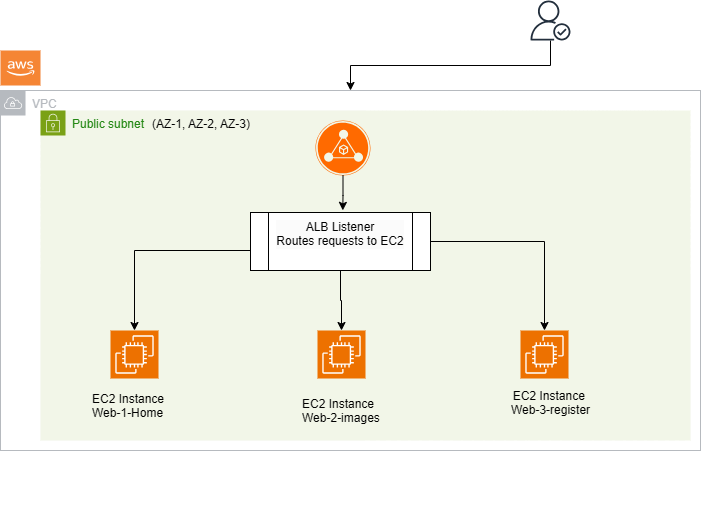
**Use Case:1: Setting Up ALB and Listener Rules for Different Request Paths**

**User**



**Project Summary:**

This project focuses on setting up an **Application Load Balancer (ALB)** to handle incoming web traffic and route it to different EC2 instances based on the requested path. The idea is to have the ALB forward traffic to the appropriate server depending on what the user wants to access on the website.

**For example:**

* **/home → Displays the homepage of the site.**
* **/images → Shows an image Page**
* **/register → Displays the register page**
* The EC2 instances are spread across multiple Availability Zones to ensure the application is highly available and can handle more users.
* All the infrastructure is managed using Terraform (which automates the creation and management of resources like EC2, ALB, subnets, etc.).
* GitHub Actions are used to automatically trigger deployments every time a change is made to the code, ensuring the infrastructure is always up-to-date and easily manageable.
* To ensure the infrastructure is always in sync and consistent, S3 is used to store Terraform state files, and versioning.

**Tech stack used and Their Purposes:**

•**Terraform:** A tool that automatically creates and manages the AWS infrastructure based on code (Infrastructure as Code). It simplifies the process of deploying and managing resources like EC2, ALB, and networking components.

•**Application Load Balancer (ALB):** The ALB is responsible for receiving user requests and routing traffic to the correct EC2 instance based on the requested path. For example, requests to /home go to one server, and requests to /register go to another. It makes the website more organized and efficient by distributing traffic based on paths.

•**EC2 Instances:** These are virtual servers that run the application code. We have three EC2 instances in this setup, each handling a different section of the site (homepage, image, registration). They are placed in different Availability Zones for better availability and fault tolerance.

•**VPC (Virtual Private Cloud):** A private network where all your AWS resources are set up. The VPC helps isolate the resources from the public internet and provides full control over your networking environment.

* Public & Private Subnets:
* Public subnets are used for the ALB since it needs to be publicly accessible.
* Private subnets are used for the EC2 instances, keeping them safe from direct internet access and allowing only the ALB to communicate with them.

•**Security Groups:** Security groups act as firewalls for your EC2 instances. They control which traffic is allowed to reach your servers. For example, only the ALB is allowed to send traffic to your EC2 instances.

•**IAM Roles & Policies:** IAM roles and policies define who can access AWS resources and what actions they can perform. For example, terraform needs permissions to create resources, and the EC2 instances need permissions to access other services like S3.

•**Route Tables:** Route tables control the movement of traffic between subnets. They define how traffic flows within the VPC, such as between private and public subnets or to the internet.

•**Internet Gateway (IGW):** The IGW allows traffic to flow between the internet and your VPC. The ALB uses the IGW to send data to and from users on the internet.

•**NAT Gateway:** The NAT gateway enables the EC2 instances in private subnets to access the internet for updates, but it prevents them from being directly accessed by the public internet.

•**S3 (Simple Storage Service):** Used to store Terraform state files. The state files track all the infrastructure and help ensure consistency. This means that everyone working on the project is using the same setup and version of the infrastructure.

•**GitHub Actions:** GitHub Actions automates the process of deploying the infrastructure and application code. Whenever there are changes in the repository, the infrastructure is automatically updated.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How the ALB and Traffic Routing Works:**

The ALB listens for incoming requests to the website and forwards them to the right EC2 instance based on the path in the URL. Here's an example of how this works:

**1. A user requests https://<dns.com>/home:**

* The ALB checks the URL path (/home), and forwards the traffic to EC2 Instance 1 (which hosts the homepage).

**2. A user requests https:// <dns.com>/images:**

* The ALB forwards the traffic to EC2 Instance 2, which handles image requests.

**3. A user requests https:// <dns.com>/register:**

* The ALB forwards the traffic to EC2 Instance 3, where the registration form is served.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GitHub URL: https://github.com/balu2422/ALB-project.git

**Conclusion:**

This project demonstrates how to use AWS services to create a reliable, scalable, and secure infrastructure that can handle traffic routing based on URL paths. By using Terraform, GitHub Actions, and other AWS services, the setup is automated, secure, and easy to manage.

**Output:**

