High - Availability Web Application on AWS Route 53.

This project showcases my skills in key AWS services including:

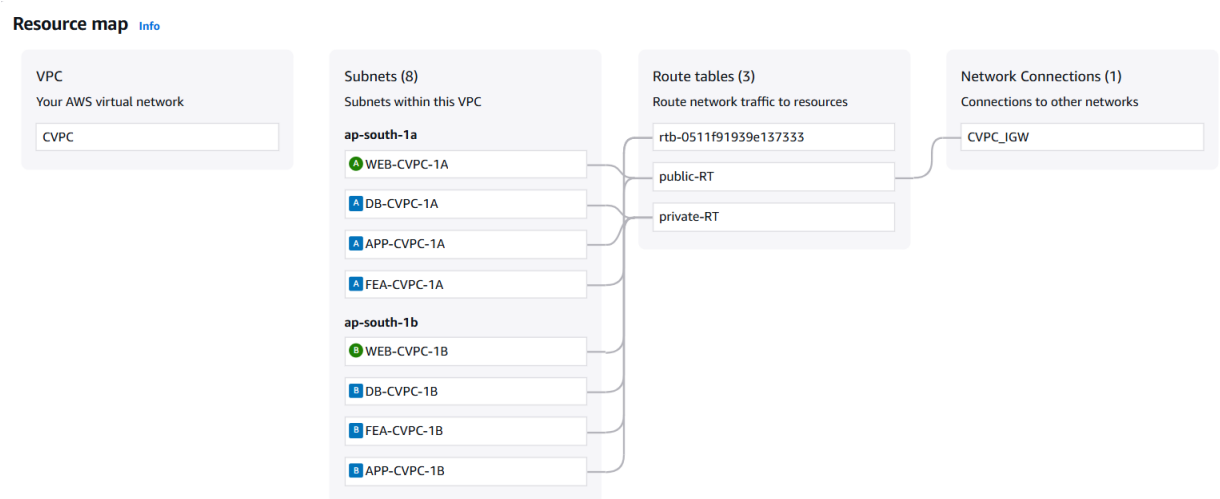
Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

AWS Services Used in the Project

- **EC2:** Running and managing virtual servers to host the web application.
- **ALB (Application Load Balancer):** Distributing incoming application traffic across multiple EC2 instances for high availability and load balancing.
- **Route 53:** Managing DNS and routing traffic to the correct endpoint using a custom domain (`ursamigo.shop`).
- **Security Groups:** Acting as virtual firewalls to control inbound and outbound traffic to and from EC2 instances and other AWS resources.
- **VPC (Virtual Private Cloud):** Providing an isolated, secure network environment where all AWS resources are launched and managed.
- **EFS (Elastic File System):** Offering a shared, scalable file system mounted across multiple EC2 instances for consistent storage of application data and user uploads.

Amazon VPC (Virtual Private Cloud)

- **Core Function:** The VPC is the foundational service for networking in AWS. It provides a logically isolated section of the AWS cloud where you can launch AWS resources. Think of it as your own private data center within AWS. This isolation gives you complete control over your virtual networking environment, including the IP address range, subnets, route tables, and network gateways.
- **Role in This Project:**
  - **Network Isolation:** All resources—EC2 instances, ALB, etc.—are contained within this VPC, ensuring they are not exposed to the public internet unless you explicitly allow it.
  - **Custom IP Range:** You defined a custom CIDR block of `198.162.0.0/23`. This gives you a specific range of IP addresses to use for your resources, preventing conflicts and allowing for organized network planning.
  - **Subnetting:** The VPC is divided into six subnets (two public and four private). This segmentation is a key security and architectural best practice.
    - **Public Subnets:** These subnets are connected to an **Internet Gateway (IGW)**, allowing resources within them to have direct internet access. Your public-facing components, like the **Application Load Balancer (ALB)** and the EC2 instances, reside here.
    - **Private Subnets:** These subnets are isolated from the internet. They are typically used for backend services like a database (e.g., RDS) that should never be publicly accessible.



Amazon EC2 (Elastic Compute Cloud)

Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

- **Core Function:** EC2 provides scalable computing capacity in the cloud. It allows you to rent virtual servers, known as **EC2 instances**, on which you can run your applications. EC2 gives you full control over the operating system, storage, and networking of these servers.
- **Role in This Project:**
    - **Application Hosting:** The two `t2.micro` instances serve as the web servers, running the application code.
    - **High Availability:** By placing one instance in `ap-south-1a` and the other in `ap-south-1b`, you have a multi-Availability Zone (AZ) deployment. If one AZ experiences an outage, the other instance can continue to serve traffic, ensuring a highly available service.
    - **Target of Load Balancer:** The EC2 instances are registered as targets in a target group, making them endpoints for the ALB to send traffic to.

🔧 Role in This Project:

1. Application Hosting

- Two **t2.micro EC2 instances** are used as **web servers**.
- Each runs your **application code**, web server (e.g., Apache/Nginx), and handles user traffic.

2. High Availability

- One instance in **`ap-south-1a`**, another in **`ap-south-1b`** (different AZs).
- Ensures **fault tolerance**—if one AZ fails, the other still serves traffic.

3. Target of Load Balancer

- Both EC2 instances are part of a **Target Group** in an **Application Load Balancer (ALB)**.
- ALB automatically sends incoming traffic to healthy EC2 instances.

🛠️ Basic Setup Steps:

1. Launch EC2 Instances

- Go to EC2 Console → "Launch Instance".
- Select **Amazon Linux 2** (or your preferred OS).
- Choose instance type: **t2.micro (free tier eligible)**.
- Select **VPC** and **AZs**: launch one in `ap-south-1a`, the other in `ap-south-1b`.
- Configure security group to allow:
    - SSH (port 22)
    - HTTP (port 80) or HTTPS (443)

2. Install Web Server (Example: Apache)

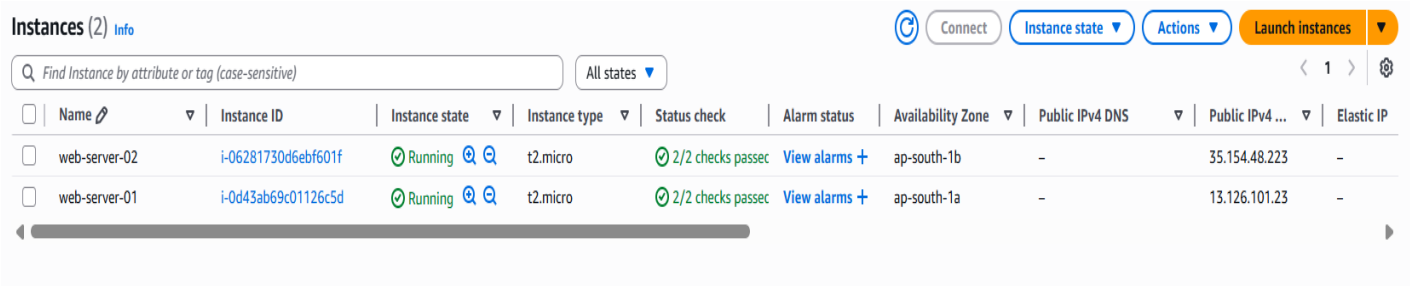
```
# Amazon Linux
sudo yum update -y
sudo yum install -y httpd
sudo systemctl start httpd
sudo systemctl enable httpd
```

3. Connect to EFS (if used)

- Mount shared EFS storage at a common path like `/mnt/efs`.

4. Register in Target Group

- Create a **Target Group**.
- Register both EC2 instances.
- Attach the Target Group to your **ALB**.



Amazon EFS (Elastic File System)

- **Core Function:** EFS provides a scalable, shared file system for use with EC2 instances. Unlike EBS, which can only be attached to a single instance (with some exceptions), EFS allows multiple EC2 instances to access the same file system at the same time.

🔧 Steps to Set Up EFS:

1. Create EFS:

- Go to AWS Console → **EFS** → "Create File System".
- Choose the **same VPC** as your EC2s.
- Leave defaults (General Purpose, Bursting).
- Click **Create**.

2. Security Groups:

- Allow **NFS (TCP 2049)** from EC2's security group to EFS.

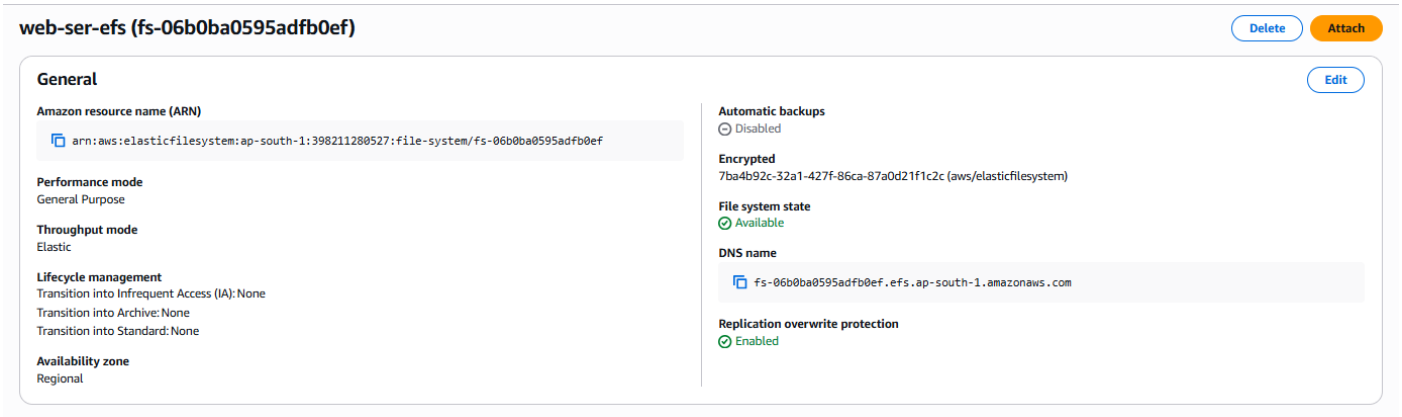
3. Install NFS Client on EC2:

```
# Amazon Linux
sudo yum install -y nfs-utils

# Ubuntu
sudo apt-get install -y nfs-common
```

4. Mount EFS on EC2:

```
sudo mkdir /mnt/efs
sudo mount -t nfs4 -o nfsvers=4.1 fs-xxxx.efs.<region>.amazonaws.com:/ /mnt/efs
```



---

5. Application Load Balancer (ALB)

- **Core Function:** An ALB is a highly available service that automatically distributes incoming application traffic to multiple targets, such as EC2 instances. It operates at the application layer and can perform sophisticated routing based on the content of the request (e.g., host-based or path-based routing).
- **Role in This Project:**
  - **Single Entry Point:** The ALB's DNS name acts as a single, public-facing endpoint for the application, abstracting away the multiple EC2 instances in the backend.
  - **Traffic Distribution:** It evenly distributes HTTP traffic across the two EC2 instances to prevent any single server from becoming overloaded.
  - **Health Checks:** The ALB continuously performs health checks on the instances and only sends traffic to those that are healthy and responsive.

🛠️ Basic Setup Steps:

1. Create Target Group

- Go to EC2 Console → Load Balancers → Target Groups.
- Create a **Target Group** (HTTP).
- Register your **two EC2 instances**.
- Set up **health check path** (e.g., `/health` or `/`).

2. Create Application Load Balancer

- Go to EC2 Console → Load Balancers → “Create Load Balancer”.
- Choose **Application Load Balancer**.
- Select:
  - **Internet-facing** (public access)
  - **HTTP** or **HTTPS**
  - VPC and at least **2 subnets** (1 in each AZ).
- Attach the previously created **Target Group**.

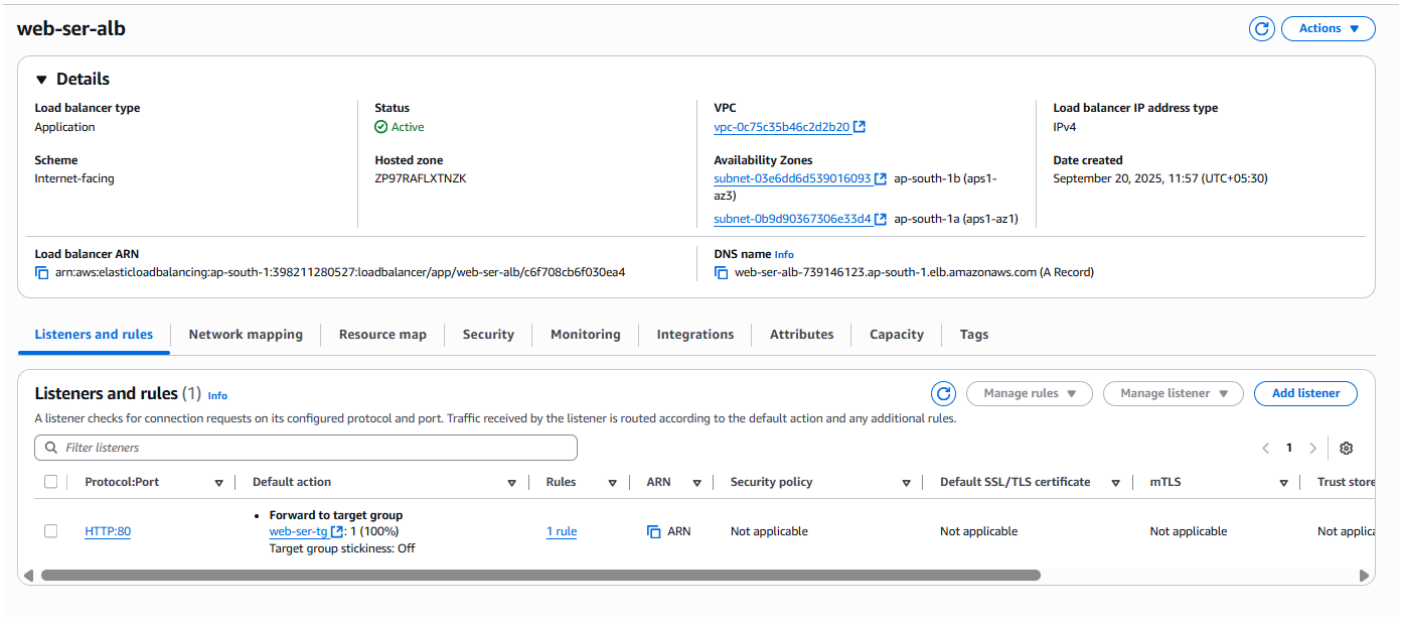
3. Configure Security Groups

- Allow **HTTP (port 80)** or **HTTPS (443)** from anywhere (or restrict as needed).

Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

4. Test ALB

- Open the **ALB DNS URL** in a browser.
- Traffic should be routed to your EC2s.
- You can check routing & health status in the **Target Group** dashboard.



Security Groups (SG)

- **Core Function:** A Security Group acts as a stateful virtual firewall for your EC2 instances and other resources. You define inbound and outbound rules to control the type of traffic that is allowed to reach or leave your resources.
- **Role in This Project:**
  - **Layered Security:** Two security groups were configured to create a layered security model:
    - **`ALB-sg`:** This security group is attached to the ALB and has an inbound rule allowing HTTP traffic (Port 80) from anywhere (`0.0.0.0/0`). This allows public internet access to the load balancer.
    - **`web-ser-sg`:** This security group is attached to the EC2 instances. Its inbound rule allows HTTP traffic (Port 80) only from the `ALB-sg`. This ensures that the web servers cannot be accessed directly from the internet, drastically reducing the attack surface.

🔧 Role in This Project:

You're using a **layered security model** with **three security groups**:

Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

---

1. 🔹 ALB-sg *(For Load Balancer)*

- **Attached to**: Application Load Balancer (ALB)
- **Inbound Rules**:
  - Allow **HTTP (Port 80)** from **anywhere** (`0.0.0.0/0`)
- **Purpose**:
  - Lets public users access the application via ALB.
  - Acts as the **main public-facing entry point**.

---

2. 🔹 web-ser-sg *(For EC2 Instances)*

- **Attached to**: EC2 Instances (web servers)
- **Inbound Rules**:
  - Allow **HTTP (Port 80)** only from **ALB-sg**
- **Purpose**:
  - Blocks direct internet access to EC2s.
  - Only the ALB can route traffic to EC2.
  - **Reduces attack surface** for better security.

---

3. 🔹 efs-sg *(For Amazon EFS)*

- **Attached to**: EFS mount targets
- **Inbound Rules**:
  - Allow **NFS (Port 2049)** only from **web-ser-sg**
- **Purpose**:
  - Ensures **only EC2 instances** can access the EFS file system.
  - Protects shared file storage from unauthorized access.

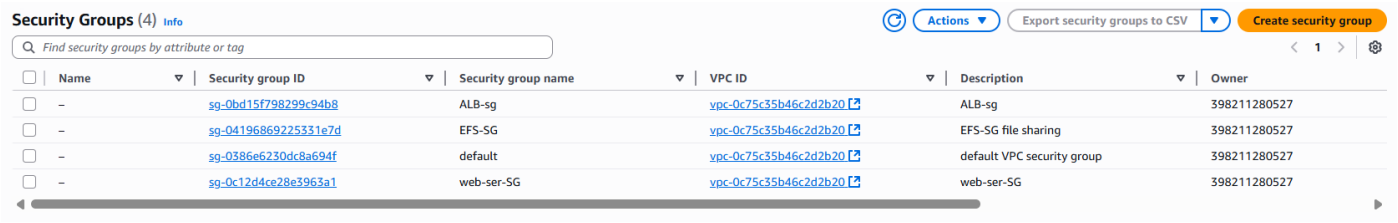
🛠️ Summary of SG Rules:

SG Name	Attached To	Inbound Rule(s)	Source
ALB-sg	ALB	HTTP (80) from `0.0.0.0/0`	Public Internet
web-ser-sg	EC2 Instances	HTTP (80) from `ALB-sg`	Internal (ALB only)
efs-sg	EFS Mount Targets	NFS (2049) from `web-ser-sg`	Internal (EC2 only)

👷 Result:

Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

- Public traffic → ALB → EC2s (web servers) → EFS (shared storage).
- No direct access to EC2 or EFS from outside.
- **Secure, layered architecture** with **controlled traffic flow**.



---

7. Target Groups (TG)

- **Core Function:** A Target Group is a logical grouping of targets (e.g., EC2 instances, IP addresses) used by a load balancer. It enables the load balancer to route requests to specific targets.
- **Role in This Project:**
  - **Target Registration:** The two EC2 instances were registered with the `web-ser-tg` target group.
  - **Health Monitoring:** The target group performs ongoing health checks on the registered instances. If an instance fails a health check, the ALB will stop sending traffic to it until it becomes healthy again, which is crucial for maintaining high availability.

🛠️ Basic Setup Steps:

Create Target Group:

- Type: **Instances**
- Protocol: **HTTP**, Port: **80**
- Target Type: **instance**
- VPC: Choose same as EC2
- Health Check:
  - Protocol: **HTTP**
  - Path: `/` (or a custom health endpoint)
  - Port: **traffic port**

Register Targets:

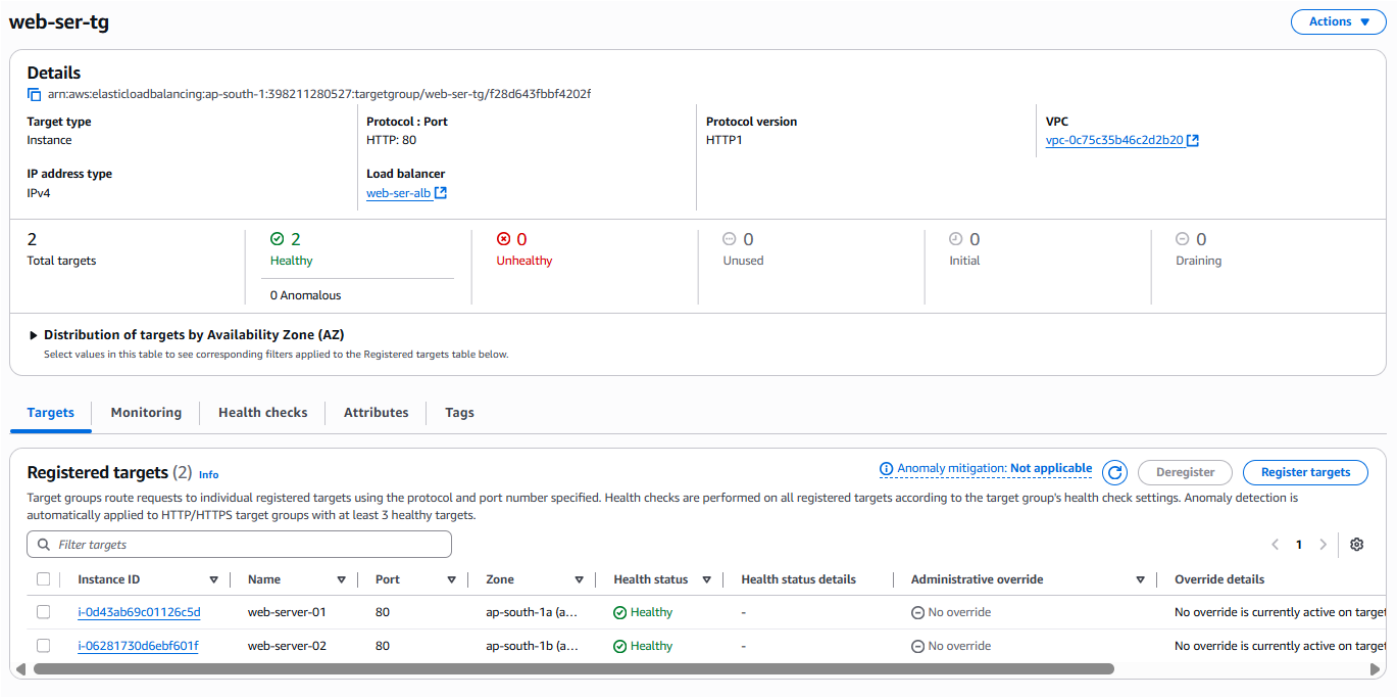
- Select your **two EC2 instances**.
- Ensure they are **in running state** and have the correct **security groups**.

Attach to ALB:

- During or after ALB creation, attach the **`web-ser-tg`** to the **listener rule** for forwarding traffic.

👷 Result:

- ALB → routes traffic to **healthy EC2s** in `web-ser-tg`.
- Auto-heals by **removing unhealthy instances** temporarily.
- Enables **reliable, scalable, and fault-tolerant** application delivery.



Amazon Route 53

- **Core Function:** Route 53 is a highly scalable and reliable Domain Name System (DNS) web service. It translates human-readable domain names (like `ursamigo.shop`) into the IP addresses or endpoints of your AWS resources.
- **Role in This Project:**
    - **Domain Mapping:** A public hosted zone was created for the domain `ursamigo.shop`. An **A record** was then configured with an **Alias** pointing to the Application Load Balancer's DNS name. This makes the application accessible to the public via the easy-to-remember domain name instead of a long, complex DNS name provided by the ALB.

🔧 Role in This Project:

Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

1. Domain Mapping

- A **public hosted zone** was created in Route 53 for the domain:
  ➤ `ursamigo.shop`

2. Alias Record to ALB

- An **A (Alias) Record** was created:
  - Name: `ursamigo.shop`
  - Type: `A - IPv4 address`
  - Alias: **Yes**
  - Alias Target: **ALB DNS name** (e.g., `myapp-alb-123456.elb.amazonaws.com`)

3. Simplified Access

- Now, users can access your app via:
  ➤ `https://ursamigo.shop`
  instead of the long ALB DNS name.
- **Improves branding**, **usability**, and **professionalism**.

🛠️ Basic Setup Steps:

☑ Step 1: Register or Use Existing Domain

- You can register `ursamigo.shop` via Route 53 or import it if registered elsewhere.

☑ Step 2: Create a Hosted Zone

- Go to Route 53 → **Hosted Zones** → **Create Hosted Zone**
  - Type: **Public Hosted Zone**
  - Domain name: `ursamigo.shop`

☑ Step 3: Create A Record (Alias to ALB)

- Record Type: **A (Alias)**
- Name: `@` (for root domain)
- Alias: **Yes**
- Target: Select your **ALB DNS name** from the dropdown

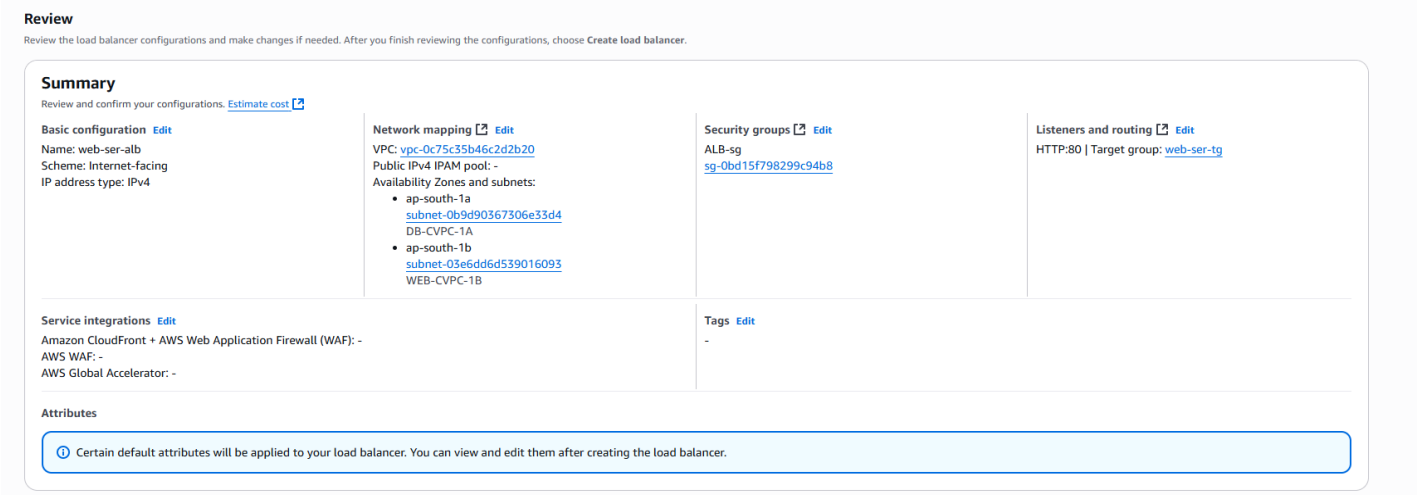
☑ Step 4: Update Domain Name Servers (If Needed)

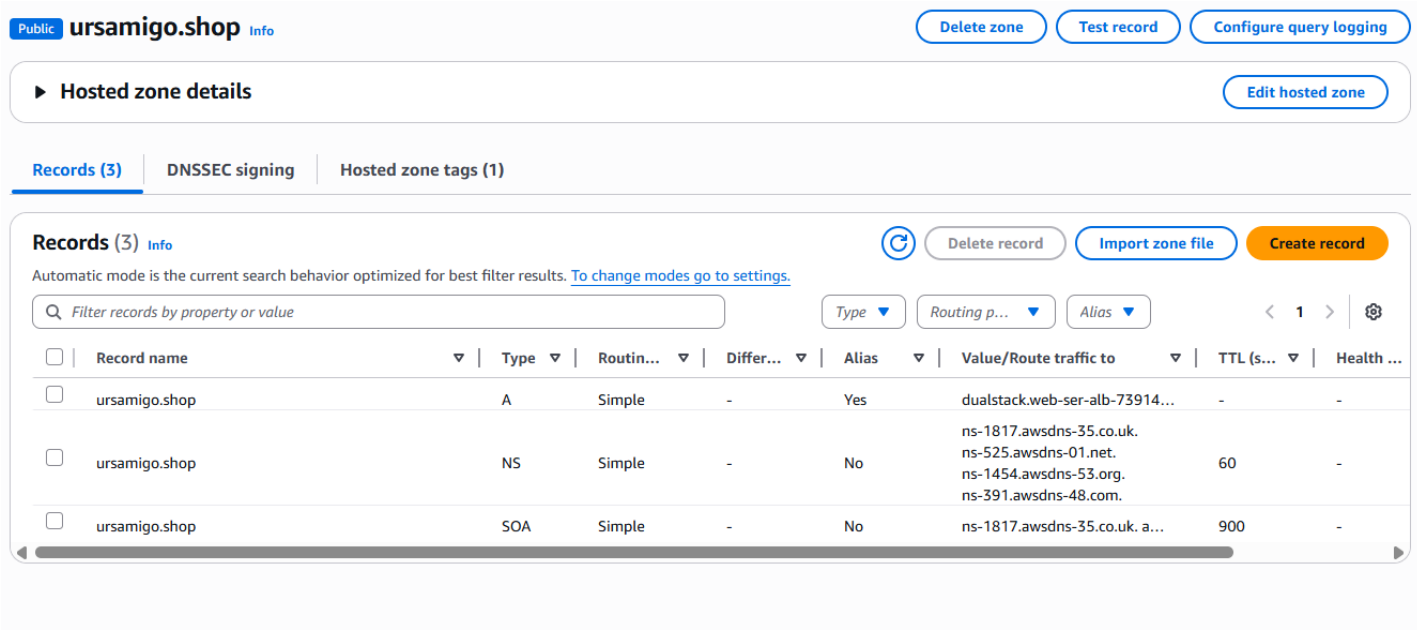
- If your domain is registered outside AWS, update its **Name Server (NS)** records to match the **NS records from Route 53**.

🤦 Result:

Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

- Users access the app via **ursamigo.shop**, which points to your **ALB**.
- DNS resolution is handled by Route 53 — fast, reliable, and globally available.
- Application stays highly available and accessible with a **custom domain**.





Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai

AVIZ ACADEMY

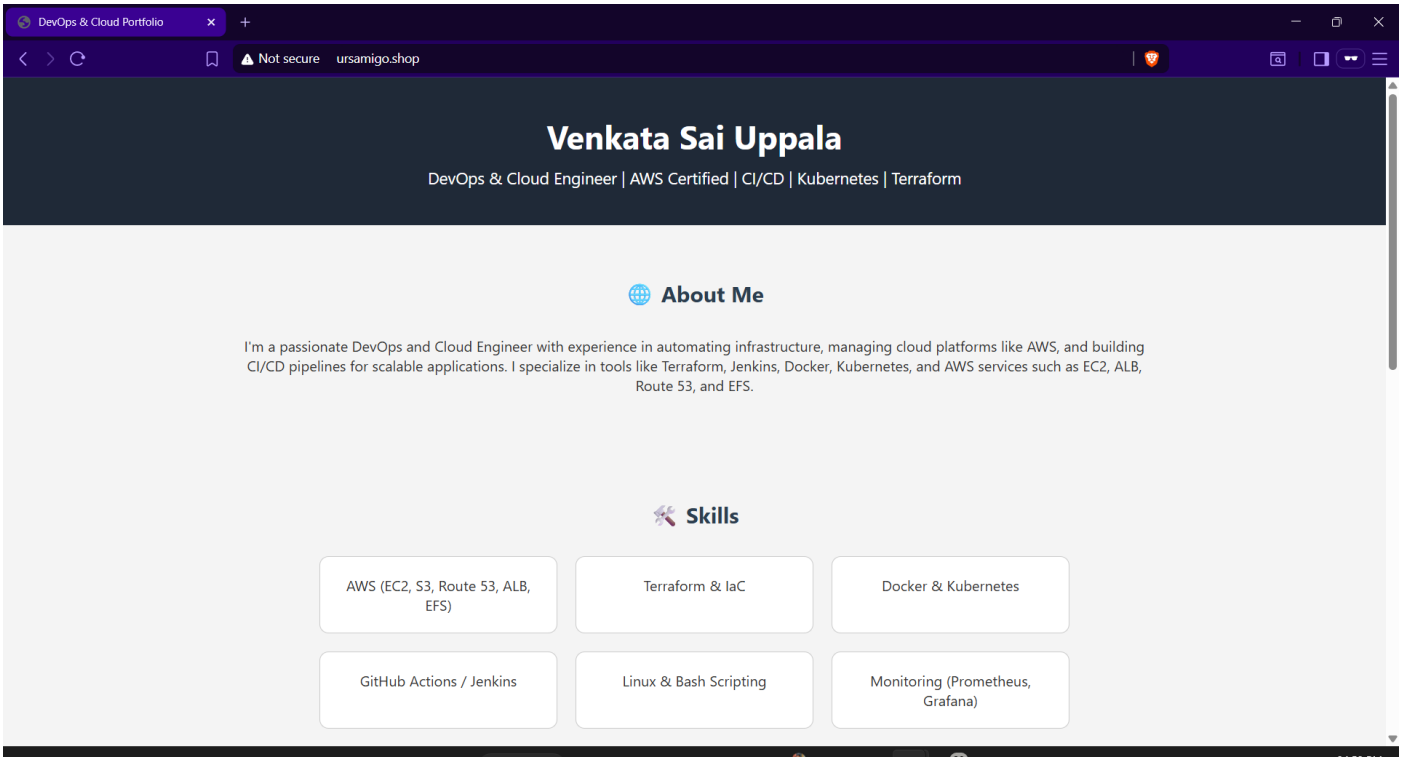
I'm excited to share that my web application is now **live** and accessible to everyone at:

🌐 **http://ursamigo.shop**

This domain is powered by **Amazon Route 53**, and traffic is managed through a **highly available AWS infrastructure**, including:

- **EC2 instances** in multiple Availability Zones
- An **Application Load Balancer (ALB)** for traffic distribution
- A **shared EFS storage** system for consistency
- Secure and optimized with **Security Groups and Target Groups**

Feel free to explore the site — more features and improvements are on the way!



Linkedin : https://www.linkedin.com/in/uppala-venkata-sai/
Github : https://github.com/UppalavenkataSai