Session 14 HA II

Lab1: SnapGallery – Highly Available Web App with Shared Storage and S3 Error Page

---

Scenario Summary

The DevOps Team has been tasked to provide infrastructure support for one of our clients running a product, **SnapGallery**, which is preparing for a major product launch. The CTO requires a resilient AWS infrastructure to host their image-sharing web application with high availability, shared storage, and graceful failure handling.

Your task is to **build a production-ready web application infrastructure** on AWS that meets the following requirements:

- Two EC2 instances (in different Availability Zones) will host the Apache-based web application and serve image content.
- Both EC2 instances must mount a **shared Amazon EFS file system** to store user-uploaded content, ensuring consistency and availability.
- An Application Load Balancer (ALB) must distribute incoming traffic evenly across the two EC2 instances.
- In the event **that both EC2 instances become unavailable**, the ALB should automatically **redirect users to a static error page hosted on Amazon S3.**

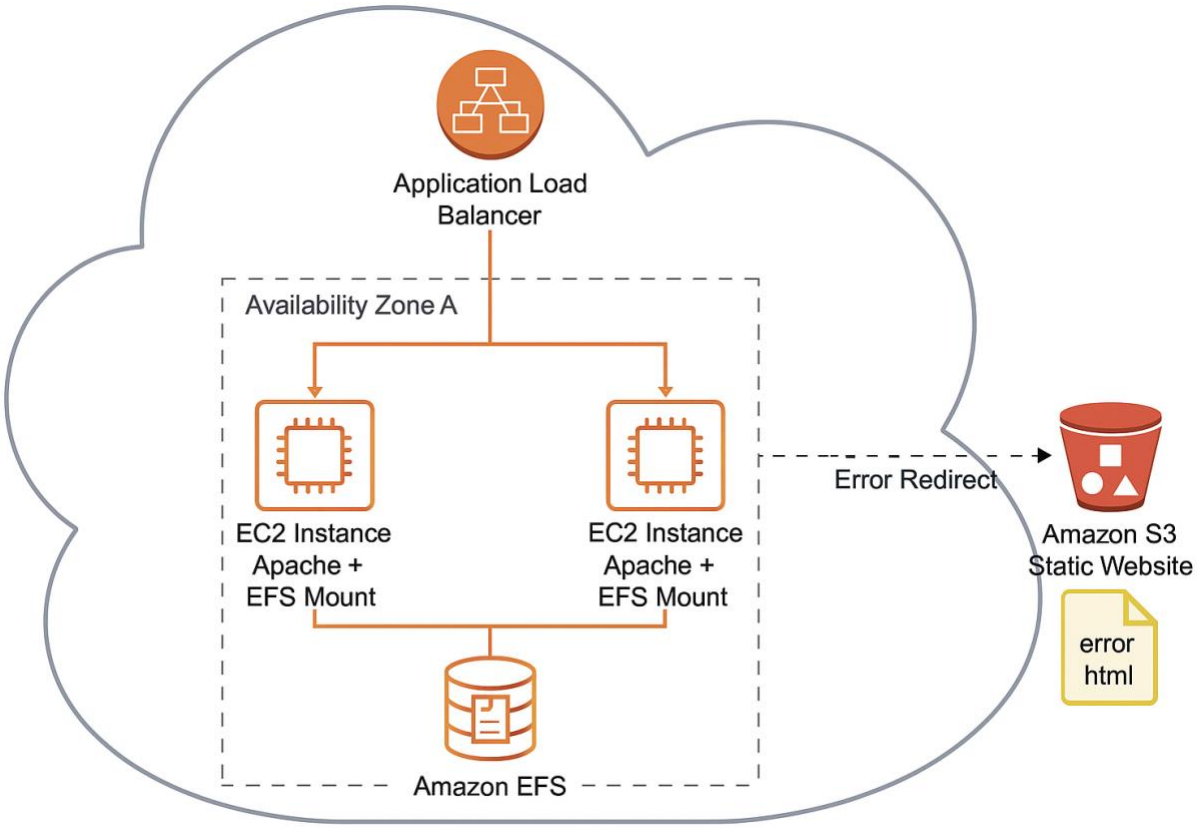
Your solution must be robust, scalable, and capable of handling production traffic, even in the face of instance failures or zone outages.

Resource Specifications

- **A Virtual Private Cloud (VPC) with at least two public subnets located in different Availability Zones.**
- **Two Amazon EC2 instances running Amazon Linux 23 or Ubuntu 20.04, each with Apache installed and configured to serve web content.**
- **A shared Amazon EFS file system with mount targets in both Availability Zones and mounted to both EC2 instances.**
- **Security groups allowing HTTP access on port 80 from the Application Load Balancer, NFS access for EFS mounting, and SSH access from your IP.**
- **An Application Load Balancer with a target group containing both EC2 instances and configured health checks.**

- **A static S3 bucket with public access enabled for static website hosting and containing a custom error.html page.**
- **A listener rule in the ALB to redirect traffic to the S3 static website when the target group is unavailable.**
- **(Optional) An IAM role attached to EC2 instances for SSM access and limited S3 permissions, if needed for future automation.**

Architecture Diagram



Section 1: Create Network Infrastructure

✅ Objective:

Set up a custom VPC with two public subnets across multiple AZs, internet access, and routing.

Steps:

1. **Create a VPC**
   - Name: `snapgallery-vpc`
   - CIDR: `10.0.0.0/16`
   - Enable DNS hostnames
2. **Create Subnets**
   - `Public-Subnet-1` (AZ-a): `10.0.1.0/24`
   - `Public-Subnet-2` (AZ-b): `10.0.2.0/24`
3. **Create and Attach an Internet Gateway**
   - Name: `snapgallery-igw`
   - Attach it to `snapgallery-vpc`
4. **Route Table**
   - Name: `public-rt`
   - Route: `0.0.0.0/0 → Internet Gateway`
   - Associate with both public subnets
5. **Enable Auto-assign Public IP**
   - For both subnets: Actions → Modify auto-assign IP → Enable

---

Section 2: Create Security Groups

✅ Objective:

Allow traffic to/from EC2, EFS, and ALB securely.

Steps:

1. **Create Security Group: `sg-web-efs`**
   - Inbound rules:
     - HTTP (80) from 0.0.0.0/0
     - SSH (22) from your IP
     - NFS (2049) from self (sg-web-efs)
   - Outbound: All traffic
2. **Create Security Group: `sg-alb`**
   - Inbound rules:
     - HTTP (80) from 0.0.0.0/0
   - Outbound: All traffic

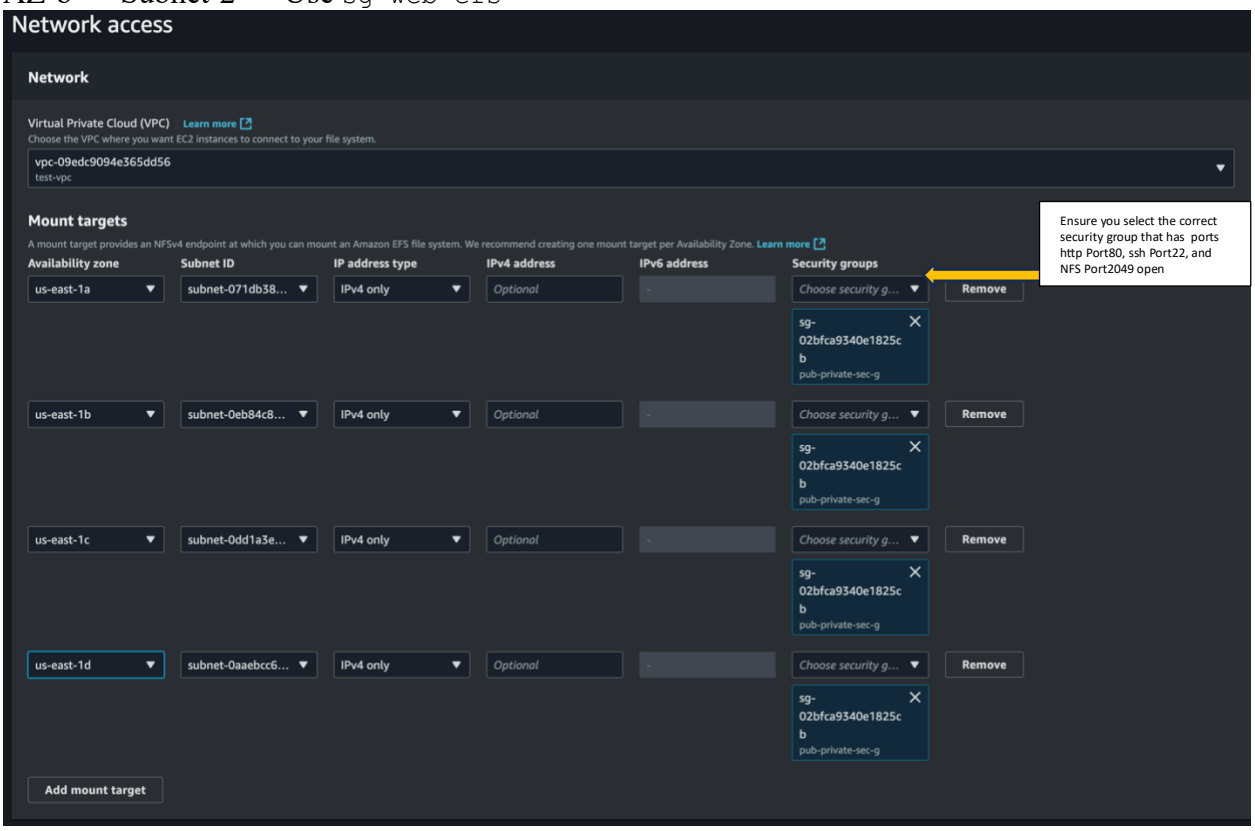
Section 3: Create EFS File System

✅ Objective:

Provision a shared file system accessible by both EC2 instances.

Steps:

1. Go to **EFS → Create File System**
   - Name: `snapgallery-efs`
   - VPC: `snapgallery-vpc`
2. **Add Mount Targets**
   - AZ-a → Subnet-1 → Use `sg-web-efs`
   - AZ-b → Subnet-2 → Use `sg-web-efs`



3. Leave performance and throughput settings as default.
4. Click **Create File System** and **note the EFS ID**.

Section 4: Launch EC2 Instances

✅ Objective:

Launch **two** instances in different AZs, install Apache, and mount EFS.

Step 1:

Repeat these for **two instances**, adjusting AZ and subnet.

- AMI: **Amazon Linux 2023**
- **Instance type:** `t2.micro`
- **Network:** `snapgallery-vpc`
- **Subnet:**
  - **Instance 1** → Subnet-1
  - **Instance 2** → Subnet-2
- **Auto-assign public IP**: Enabled
- **Security group:** `sg-web-efs`

Pay attention to your Network Settings

User Data Script (update `fs-xxxxxx` with your EFS ID):

Remember it's all the way down your deployment window

```
#!/bin/bash
yum install -y amazon-efs-utils httpd
mkdir -p /mnt/photos
mount -t efs <fs-xxxxxxx>:/ /mnt/photos
ln -s /mnt/photos /var/www/html/photos
echo "<h1>Welcome to SnapGallery!</h1>" > /var/www/html/index.html
systemctl start httpd
systemctl enable httpd
```

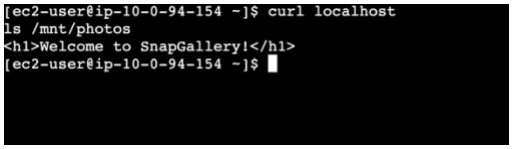
- Tag:
  - Name = `snapgallery-web-1`
  - Name = `snapgallery-web-2`

- Repeat Step1 for second Instance

🔍 Breakpoint:

- Use **Instance Connect** to verify:

  curl localhost
  ls /mnt/photos
  Output should look like this:



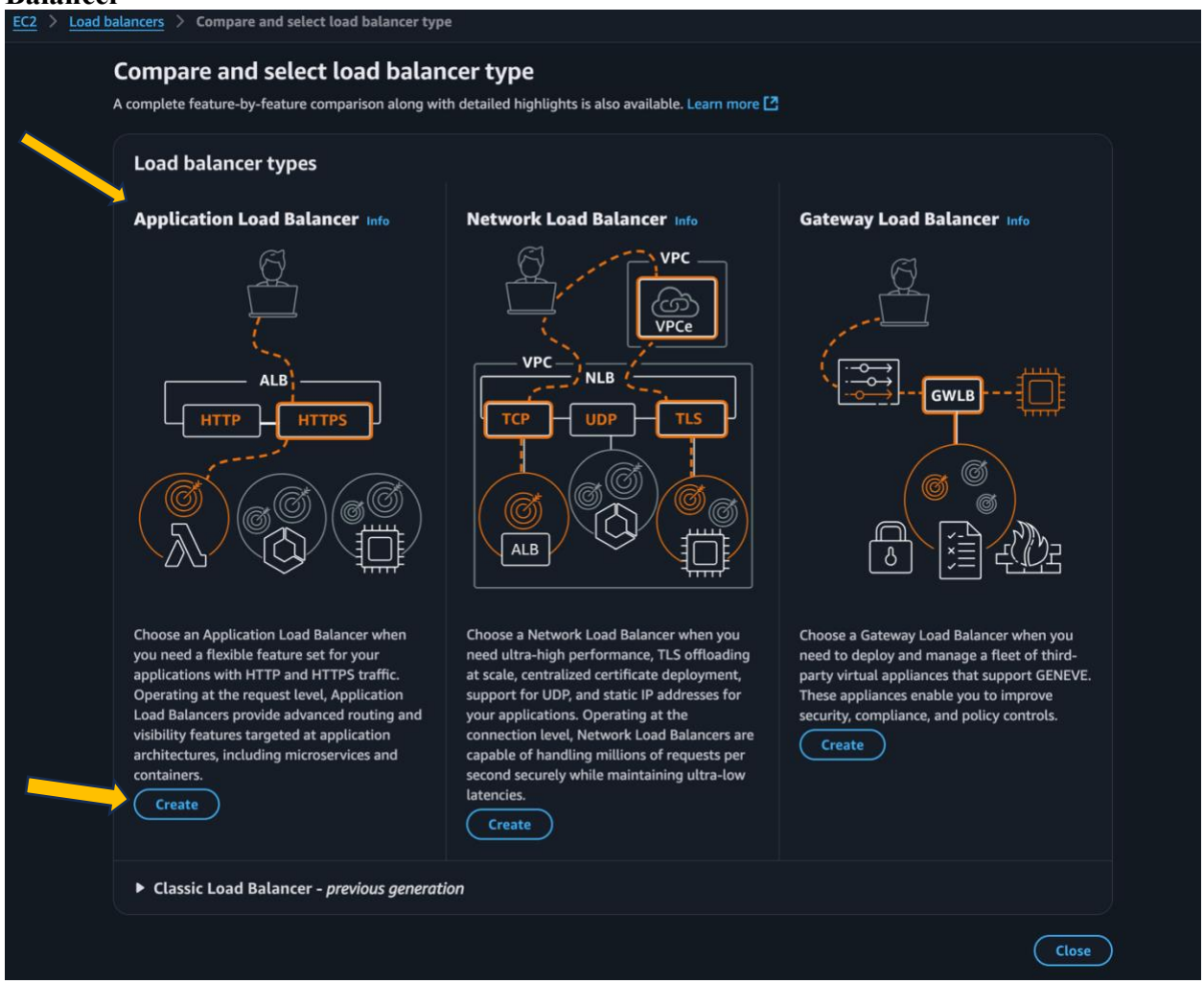
Section 5: Configure Application Load Balancer

✅ Objective:

Distribute traffic to EC2 instances and monitor health.

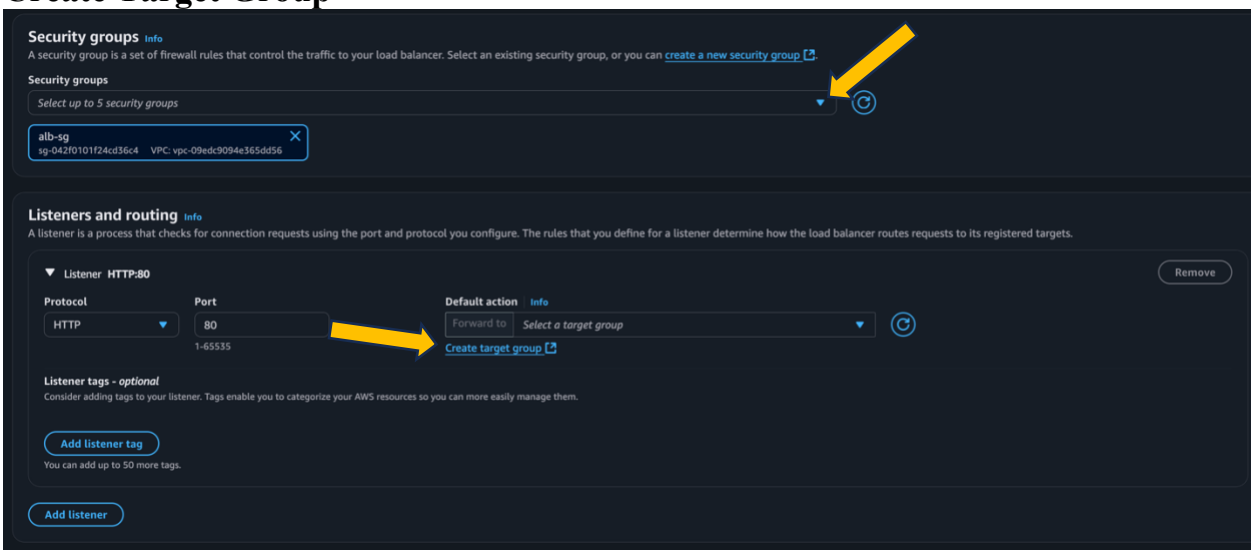
Steps:

1. Go to **EC2 → Load Balancers → Create Load Balancer → Application Load Balancer**



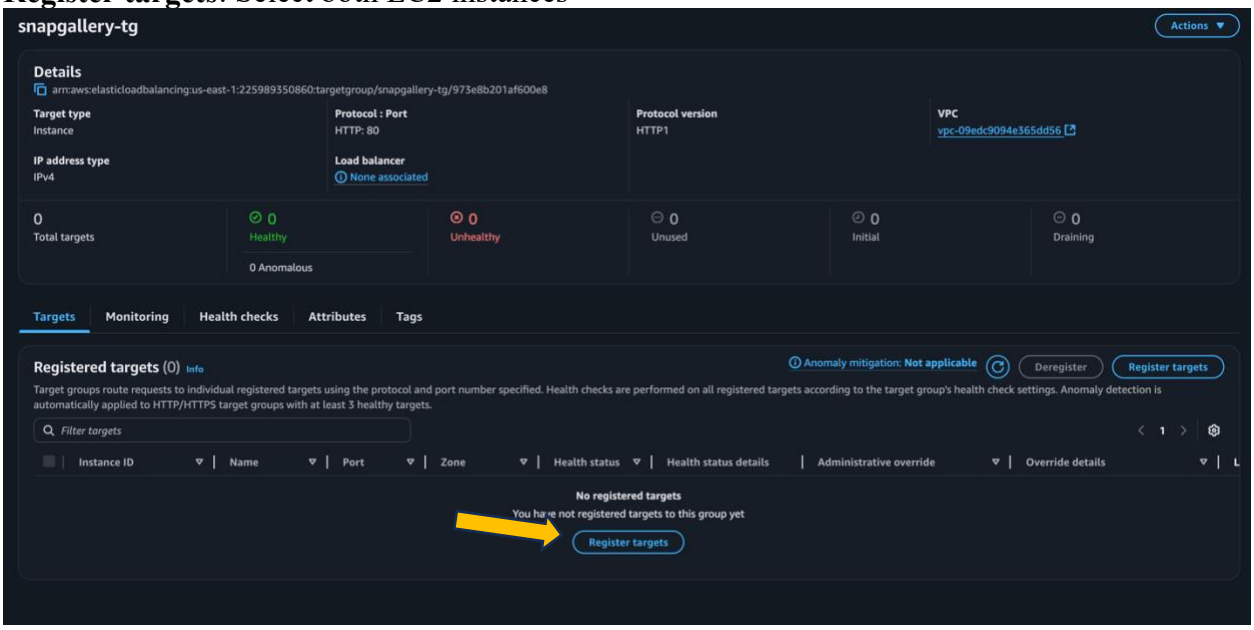
2. Name: `snapgallery-alb`
3. Scheme: Internet-facing
4. Listener: HTTP (port 80)
5. VPC: `snapgallery-vpc`
6. Subnets: Select both public subnets
7. Security group: `sg-alb`

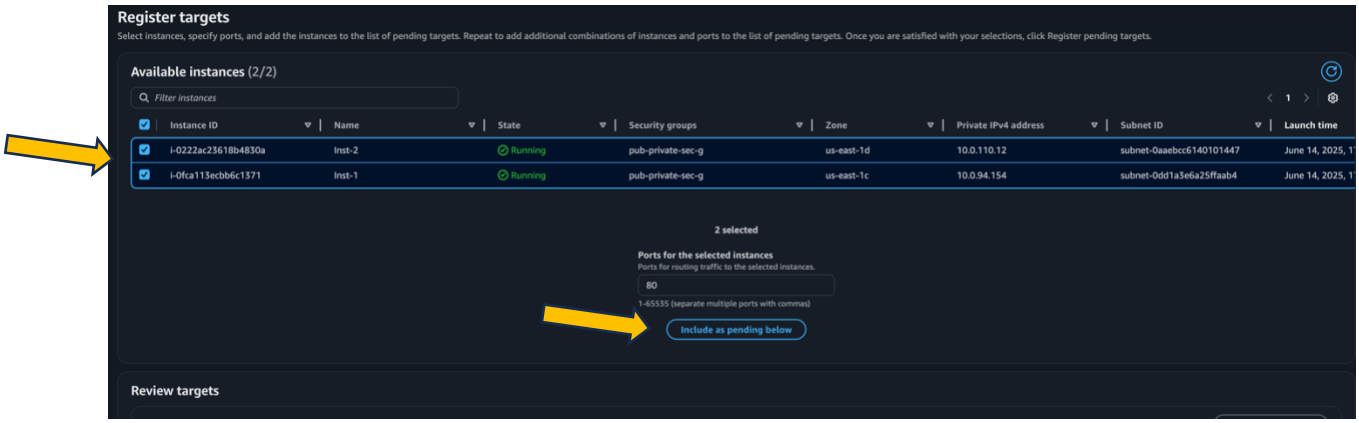
Create Target Group



- Target type: instance
- Name: `snapgallery-tg`
- Protocol: HTTP
- Port: 80
- Health check path: `/index.html`

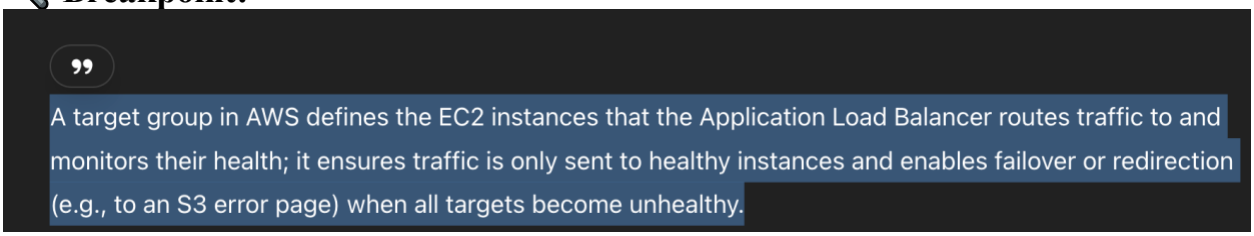
Register targets: Select both EC2 instances





Review and **register pending targets**

🔍 Breakpoint:



- ALB → Target Groups → Check health = `healthy`

⇨ **finish creating your ALB , we'll come back to it after s3**

Section 6: S3 Static Site for Error Page

✅ Objective:

Host a fallback error page in S3 to be used by ALB if all EC2s fail.

Steps:

1. Go to S3 → Create Bucket: `snapgallery-error-page`
   - Uncheck "Block all public access"

- Enable static website hosting*(Go into the created **bucket>properties>enable Staticwebhosting**)*
- Index document: `error.html`

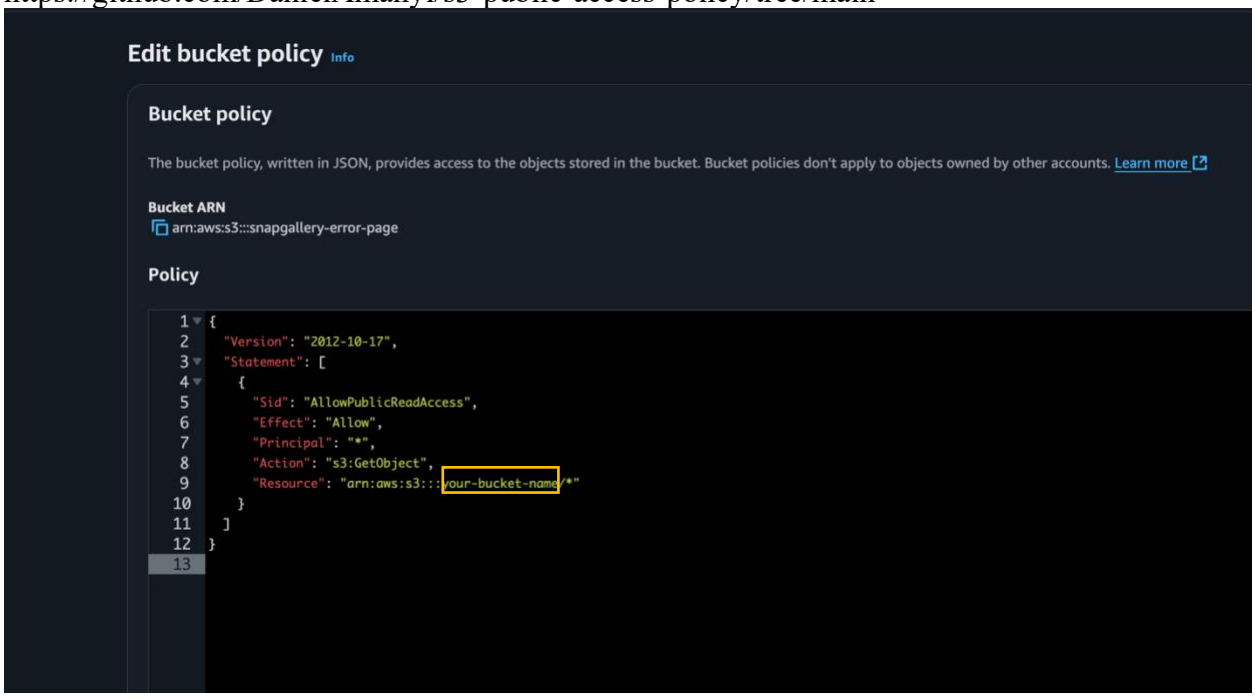
2. Upload `error.html`:

```
You can create a .html error page, or download one from
https://github.com/rapidruby/simple-error-pages?tab=readme-ov-file
```

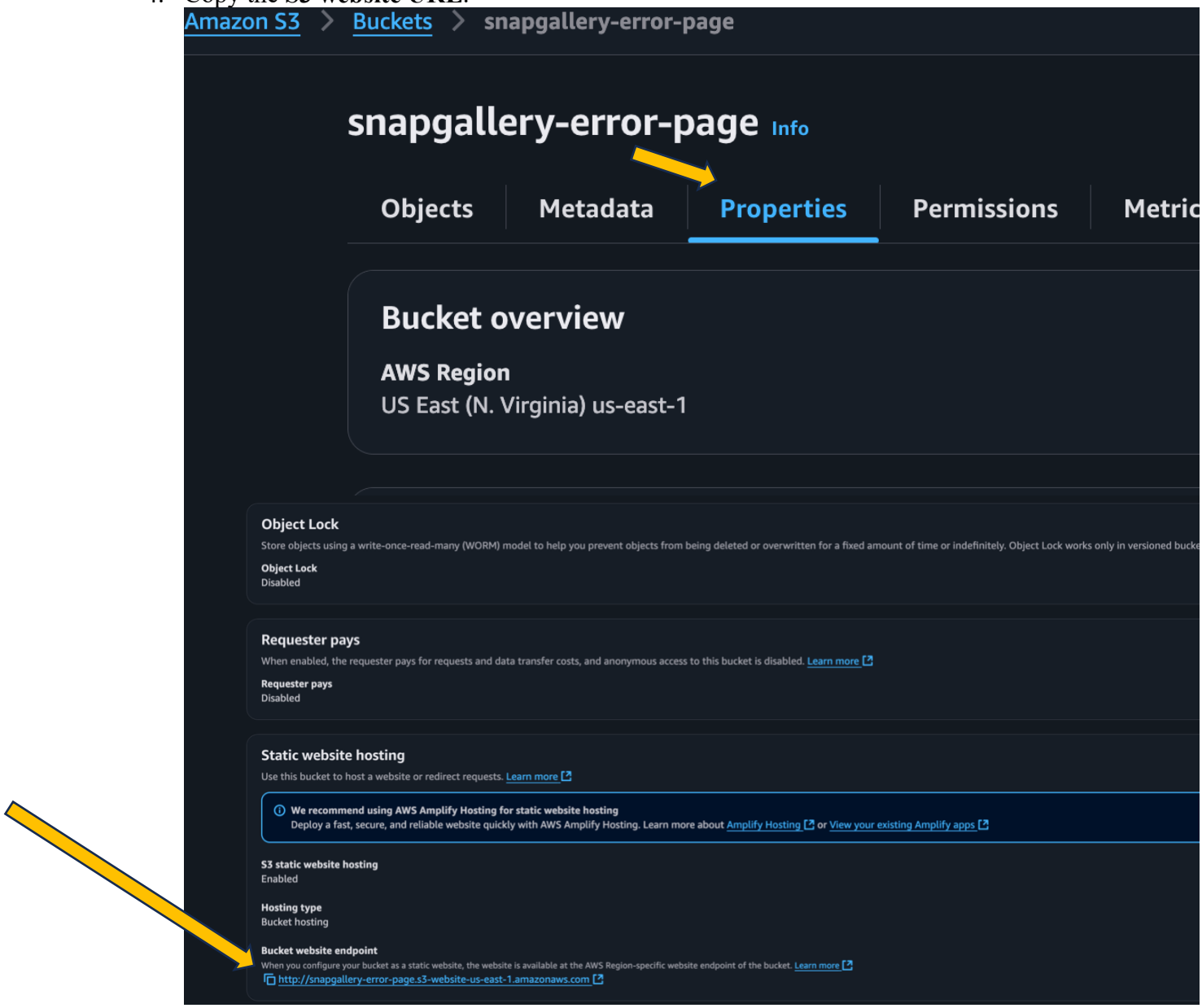
3. Set public read access:
   - Use object permissions or bucket policy

   (Remember to access through your permissions Tab)

   https://github.com/DanielAmanyi/s3-public-access-policy/tree/main



4. Copy the **S3 website URL**.



Go back to your ALB TAB

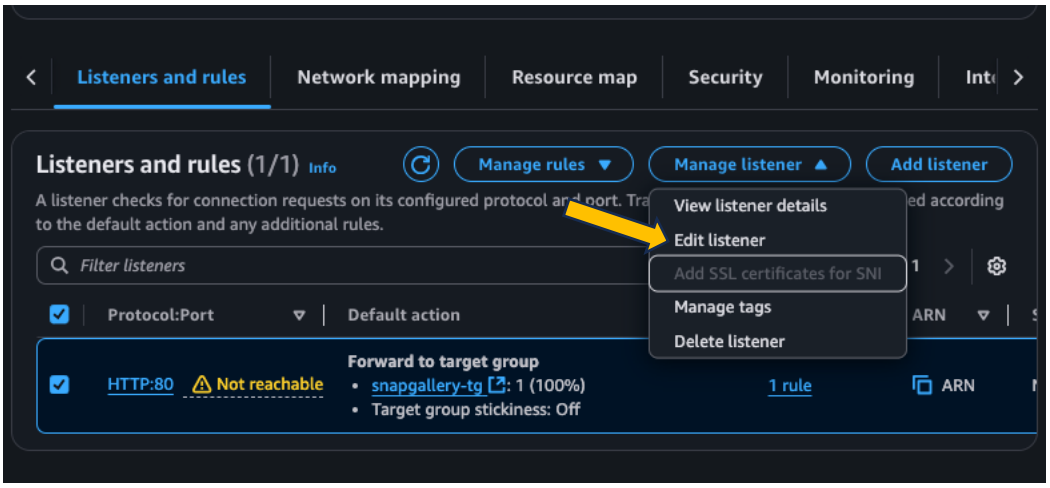
Section 7: Add ALB Error Response Redirect

✅ Objective:

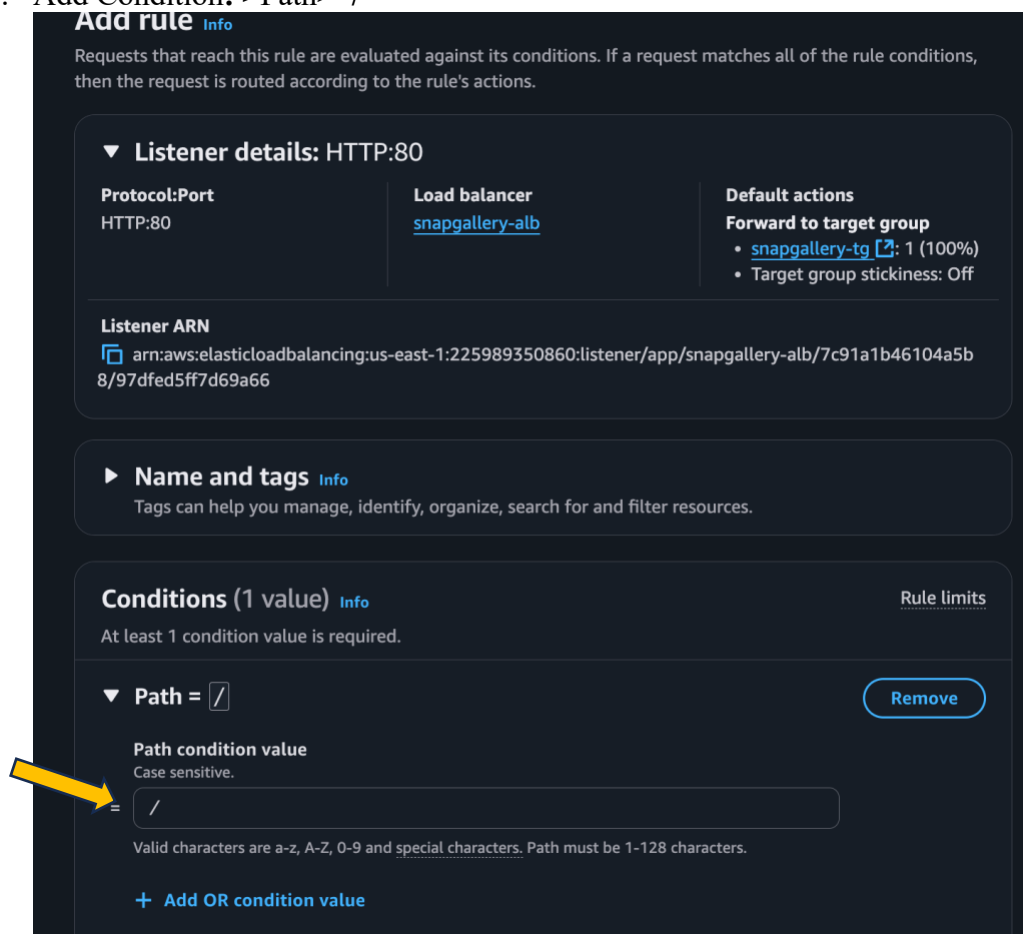
Make the ALB redirect to S3 error page when EC2s are unhealthy.

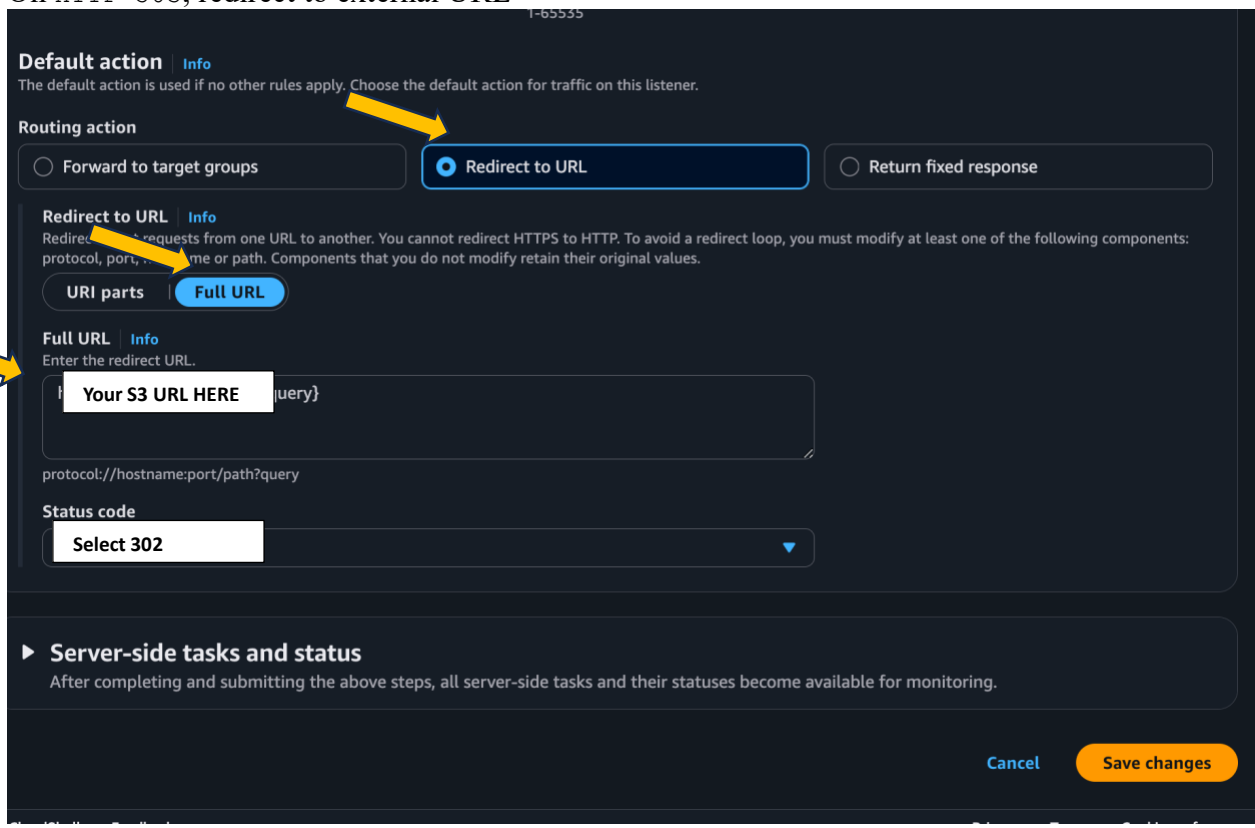
Steps:

1. EC2 → Load Balancers → Select `snapgallery-alb`
2. Go to **Listeners** → View/edit rules



3. Add Condition: >Path> /



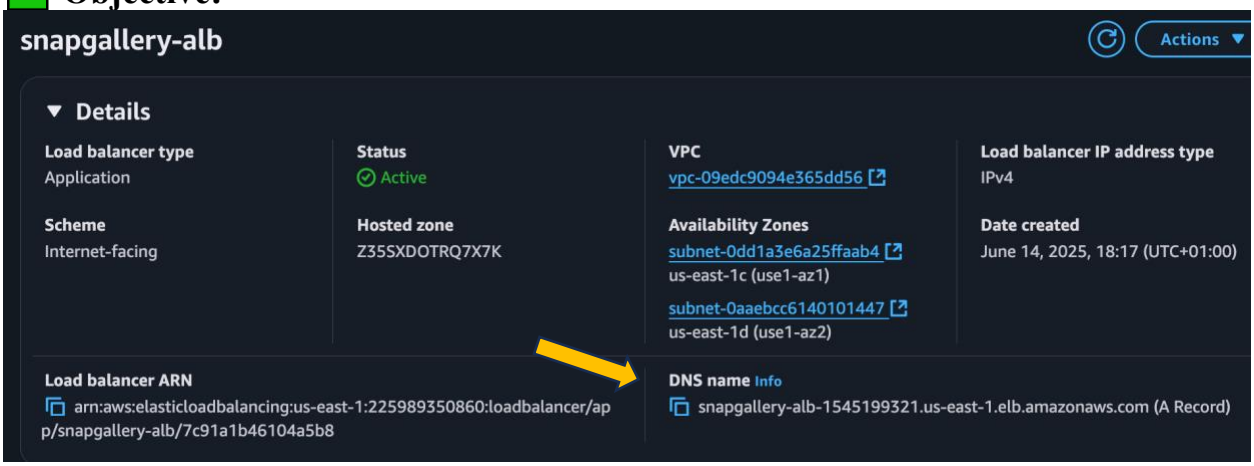
- On `HTTP 503`, redirect to external URL
- 

- Enter your S3 site URL (e.g., `http://snapgallery-error-page.s3-website-<region>.amazonaws.com/error.html`)
- Priority>**1**

Section 8: Test the Setup

Go Back to ALB

✅ Objective:



Confirm content is served, EFS is shared, and error handling works.

1. Visit the **ALB DNS name** in your browser
   - Should load `Welcome to SnapGallery!`
2. On EC2 instance:

```
echo "New photo uploaded" > /mnt/photos/test.txt
```

3. On the other EC2:

```
cat /mnt/photos/test.txt
```

4. **Simulate failure**:
   - Stop both EC2s
   - Revisit ALB URL → You should be redirected to your S3 error page

Project Objective

The objective of this project is to reinforce your hands-on capabilities in the following key areas:

- Cloud Networking
- Virtual Machine Configuration
- Network File Storage
- Data Persistence
- Object Storage
- Distributed Systems
- High Availability

- Microservices