# **AWS Load Balancer Task**

**Overall Task Objective:**

The objective of this assignment is to design, deploy, secure, and validate a complete load-balancing and traffic-routing architecture using AWS Elastic Load Balancers. This includes configuring Classic, Application, and Network Load Balancers; enabling SSL termination; mapping DNS via Route 53; and implementing full logging and monitoring by pushing ALB access logs to Amazon S3. The assignment ensures hands-on experience with high availability, scalability, secure communication, traffic routing, and operational visibility within AWS.

**Overall Task Prerequisites:**

Before beginning the assignment, the following prerequisites must be in place:

**🔹 AWS Infrastructure Requirements**

* A working **AWS Account (root user)**
* Region set to **N. Virginia (us-east-1)**
* A fully configured **VPC**
* **Public subnets** across at least 2 Availability Zones
* **Internet Gateway (IGW)** attached to the VPC
* **Route tables** properly configured

**🔹 Compute Requirements**

* At least **2 EC2 instances** for Classic LB
* EC2 instances with:
  + Security group allowing HTTP (80)
  + Apache installed (for ALB/NLB testing)
  + Health check path (/) functional

**🔹 Networking & Access Requirements**

* Security groups allowing:
  + **Port 80** (HTTP)
  + **Port 443** (HTTPS)
  + **Port 22** (SSH)
* Public IP enabled for testing
* Key pair generated for SSH access

**🔹 SSL/TLS Requirements**

* Domain registered
* Hosted Zone created in **Route 53**
* SSL certificate issued in **ACM** (DNS validation recommended)

**🔹 S3 & Logging Requirements**

* S3 bucket created for storing ALB access logs
* Proper bucket policy allowing ALB to push logs

**🔹 Route 53 Requirements**

* Domain ownership
* DNS records configured
* Ability to map ALB using Alias A/AAAA record

1. Configure Classic Load balancer.

**Task Title**

Configure Classic Load Balancer (CLB) for EC2 Instances

**Objective**

To deploy and configure a Classic Load Balancer that distributes HTTP traffic across EC2 instances.

**Prerequisites**

* VPC created
* At least **2 EC2 instances** running in different AZs
* Security group allowing **HTTP (80)**
* EC2 health check path ready
* IAM permissions for ELB
* Public subnets available

**Step-by-Step Implementation (With Evidence)**

**Step 1: Navigate to CLB Creation**

* Console: **EC2 → Load Balancers → Create Load Balancer → Classic Load Balancer**

**Step 2: Configure Basic Settings**

* Name: classic-lb
* Type: Internet-facing
* Listener: HTTP 80 → HTTP 80

**Step 3: Select Security Groups**

* Choose SG allowing port 80

**Step 4: Configure Health Check**

* Ping Target: HTTP:80/
* Healthy threshold: 2
* Unhealthy threshold: 2

**Step 5: Register EC2 Instances**

* Select both web servers
* First created one vpc

A screenshot of a computer

AI-generated content may be incorrect.

* Created Two public subnets

A screenshot of a computer

AI-generated content may be incorrect.

* Created one route table and associated two subnets to it

A screenshot of a computer

AI-generated content may be incorrect.

* Created one internet gateway and attach it to vpc
* Attach igw to rote table

A screenshot of a computer

AI-generated content may be incorrect.

* Now, launched two ec2 instances

A screenshot of a computer

AI-generated content may be incorrect.

* Connected to ec2 01instances and installed httpd

Yum update -y

Yum install httpd -y

Systemctl start httpd

Systemctl status httpd

* Created index.html

A screenshot of a computer program

AI-generated content may be incorrect.

* Using ec2 public Ip we can check it in browser by port number 80 at the end. To check whether the service working or not.

A screenshot of a computer

AI-generated content may be incorrect.

* Connected to ec2 02 instances and installed httpd

Yum update -y

Yum install httpd -y

Systemctl start httpd

Systemctl status httpd

* Created index.html

A screenshot of a computer program

AI-generated content may be incorrect.

* As we can service is working

A black rectangular object with white text

AI-generated content may be incorrect.

* Now, we are creating classic load balancer > give a name > select vpc and subnets which are in different zones > select two ec2 instances > click on create. remaining everything keeps them default

A screenshot of a computer

AI-generated content may be incorrect.

* Classic load balancer has been created

A screenshot of a computer

AI-generated content may be incorrect.

* Select load balancer which we created before > target instances > check health status. If it shows ‘in service’ then it is working

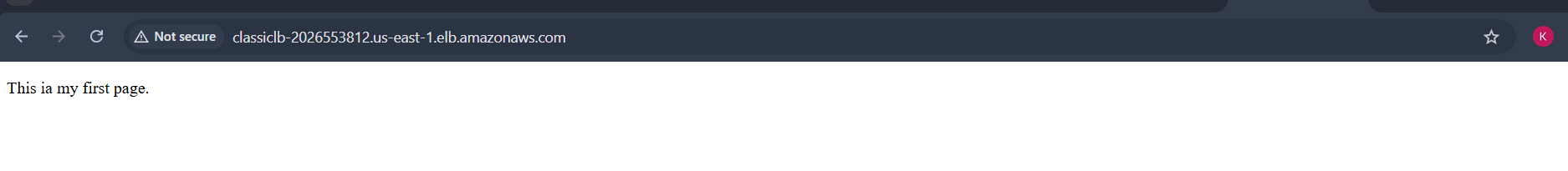
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* In details copy DNS name and run in browser
* As we can below it is working.



**Error:**

* when I first created classic load balancer, when after changing security groups it was not working. Later, I deleted it and created new one and it worked.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Validation**

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion:**

Classic Load Balancer was successfully configured and is distributing traffic across EC2 instances.

1. Configure Application Load balancer.

**configure Application Load Balancer**

**Task Title**

Deploy Application Load Balancer (ALB)

**Objective**

To configure an ALB to route HTTP traffic to EC2 instances or target groups.

**Prerequisites**

* VPC with **2 public subnets**
* EC2 instances running
* SG with port 80 open
* Target group created
* Apache installed

**Step-by-Step Implementation**

**Step 1: Navigate to ALB Creation**

Console → EC2 → Load Balancers → Create Load Balancer → Application Load Balancer

**Step 2: Configure ALB Settings**

* Name: ALBTask
* Internet-facing
* Listener: HTTP 80

**Step 3: Select Subnets**

Choose 2 public subnets in different AZ

**Step 4: Configure Security Group**

Allow port 80

**Step 5: Create Target Group**

* Type: Instance
* Health check: /

**Step 6: Register EC2 Instances**

**Step 7: Create ALB**

* **First create a target group by attaching two ec2 to it**

A screenshot of a computer

AI-generated content may be incorrect.

* **Now create application load balancer by using above configurations**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* As we can see application load balancer has been created and once status shows ‘active’. Now we can check health status of target groups

A screenshot of a computer

AI-generated content may be incorrect.

* As we can that health status is good and running for target groups.

A screenshot of a computer

AI-generated content may be incorrect.

* Copy DNS Name from created load balancer > details.
* Using URL run it in browser
* As we can be application load balancer is working successfully

A screen shot of a computer

AI-generated content may be incorrect.

**Conclusion:**

Application Load Balancer deployed successfully and routing traffic to EC2 instances.

1. Configure Network Load balancer.

**configure Network Load Balancer**

**Task Title**

Deploy Network Load Balancer (NLB)

**Objective**

To configure a high-performance NLB capable of routing TCP/UDP traffic.

**Prerequisites**

* VPC with public subnets
* EC2 instances running
* SG allowing required ports
* Target group for TCP

**Step-by-Step Implementation**

**Step 1: Create NLB**

Console → EC2 → Load Balancers → Create Load Balancer → Network Load Balancer

**Step 2: Configure Settings**

* Name: NLB01
* Scheme: Internet-facing
* Listener: TCP 80

**Step 3: Select Subnets**

Pick 2 public AZs

**Step 4: Create Target Group**

* Target type: Instance
* Protocol: TCP

**Step 5: Register Instances**

**Step 6: Create NLB**

* Creating a target group > in protocol select TCP > health checks: Http > health checks path: /index.html

A screenshot of a computer

AI-generated content may be incorrect.

* Target group has been created successfully

A screenshot of a computer

AI-generated content may be incorrect.

* Create network load balancer

A screenshot of a computer screen

AI-generated content may be incorrect.

* Using regular step as we followed before create LB
* Once created wait for status to turn active. Then check health status in target group
* Now configure security groups and add tcp and save it.

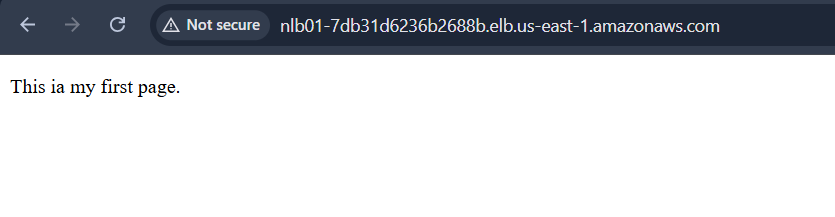
A screenshot of a computer

AI-generated content may be incorrect.

* Now, copy DNS URL from created load balancer >details here we are able see URL, using that run it in browser

**Validation:**

* As we can see network load balancer is working



**Conclusion:**

NLB was successfully configured and can handle TCP-level traffic efficiently.

1. Attach SSL for application load balancer.

**Attach SSL Certificate to Application Load Balancer**

**Task Title**

Configure SSL/TLS for ALB HTTPS Listener

**Objective**

To secure application traffic by attaching an SSL certificate to ALB using HTTPS port 443.

**Prerequisites**

* ALB already configured
* Domain name verified in ACM
* SSL certificate issued
* SG must allow 443
* Listener 443 required

**Step-by-Step Implementation**

**Step 1: Request Certificate in ACM**

* ACM → Request Public Certificate

**Step 2: Validate Domain**

* DNS validation in Route 53

**Step 3: Add HTTPS Listener to ALB**

* ALB → Listeners → Add Listener → HTTPS (443)

**Step 4: Attach SSL Certificate**

* Select ACM certificate

**Step 5: Forward to Target Group**

* Created an application load balancer. To create follow same step as before.

A screenshot of a computer

AI-generated content may be incorrect.

* It is successfully created and running

A screenshot of a computer

AI-generated content may be incorrect.

* Created target groups.
* It passed health checks

A screenshot of a computer

AI-generated content may be incorrect.

* Create an ACM certificate > Gave Domain Name >disable export > DNS Validation

A screenshot of a computer

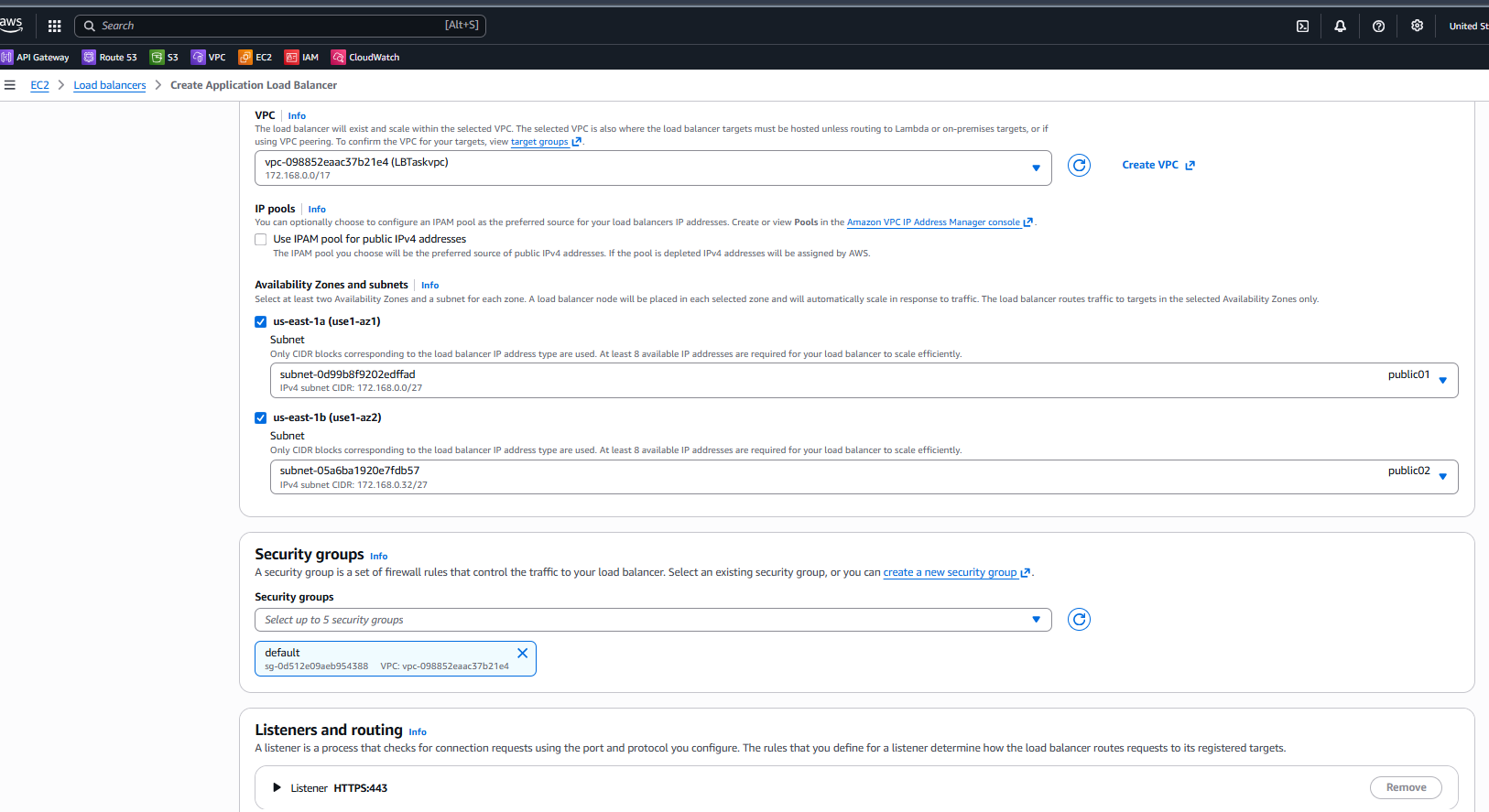
AI-generated content may be incorrect.

* ACM certificate has been created and issued. Let’s attach it to load balancer.

A screenshot of a computer

AI-generated content may be incorrect.

* Once ACM created. Configure route 53, create host zones, edit name servers in Go daddy. Detailed process is explained in next task 5
* Creating an application loader balancer



* Gave name > attached vpc, subnets, security groups > in security policy: select all security policies in drop down> certificate source: select from ACM > attach the certificate that we created in before step > keep remaining everything default > click on create
* While creating ALB in listener configure Listener 443

A screenshot of a computer

AI-generated content may be incorrect.

* As we can below application load balancer and active and we have attached ssl certificate to it
* I will take few minutes to get active status
* SG must allow 443.
* While creating ALB in listener configure Listener 443

A screenshot of a computer

AI-generated content may be incorrect.

**Issues:**

* Certificate won't validate without Route 53 DNS  
  Fix: Added correct CNAME validation record.

**Conclusion:**

SSL certificate attached successfully, and HTTPS traffic is now enabled and secure.

1. Map Application load balancer to R53.

**Map ALB to Route 53**

**Task Title**

Create Route 53 DNS Record for ALB

**Objective**

To map a domain name to the Application Load Balancer using Route 53 for easier access.

**Prerequisites**

* Registered domain
* Hosted zone in Route 53
* ALB created
* HTTPS/HTTP listener ready

**Step-by-Step Implementation**

**Step 1: Route 53 → Hosted Zones**

**Step 2: Create Record**

* Name: www.yourdomain.com
* Type: A
* Routing: Alias to ALB

**Step 3: Save Record**

* Go to route 53 > select create hosted zones > get started > Enter domain name > select public hosted zones > click on create .

A screenshot of a computer

AI-generated content may be incorrect.

* Hosted zones have been created successfully

A screenshot of a computer

AI-generated content may be incorrect.

* Once hosted zone is created. Click on it. We can see below page copy Name Server ID’s we have four theirs. Go to the website where we created our domain shoppiex. store

A screenshot of a computer

AI-generated content may be incorrect.

* Now, In Go Daddy select profile > DNS > Name Server > replace the name servers with new once which we copied before and save it.

A screenshot of a computer

AI-generated content may be incorrect.

* Redirect to route 53 > hosted zones > shoppiex. store (created hosted zone) > create record.
* Record name: (copy and paste Cname from ACM certificate which we created) and remove ‘. shoppiex. store.’ at end of link. > record type: A-Route traffic to an ipv4 address and some AWS resources. > enable Alis > Route traffic to: Alias to application and classic load balancer > select region: us north Virginia > select load balancer > routing policy: simple routing > click on create record

A screenshot of a computer

AI-generated content may be incorrect.

* Now we can Cname in hosted zone

A screenshot of a computer

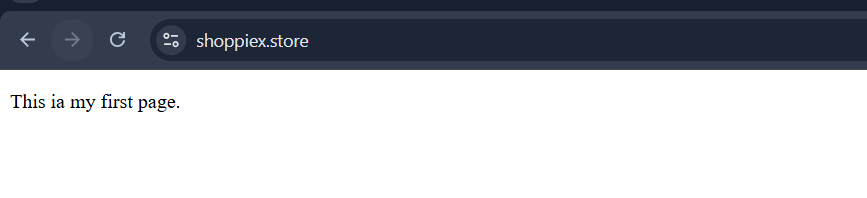
AI-generated content may be incorrect.

**Validation:**

* As we can see in below screenshot website is working. But we need to wait for few minutes to get the status
* Refresh the page for few times. It shows both pages

A close-up of a blue and white screen

AI-generated content may be incorrect.



**Conclusion:**

**Route 53 successfully mapped the domain to ALB, enabling user-friendly access.**

1. Push the application load balancer logs to S3.

**Push ALB Logs to S3**

**Task Title**

Enable ALB Access Logging to S3

**Objective**

To store detailed access logs of ALB traffic in an S3 bucket for auditing and analysis.

**Prerequisites**

* ALB created
* S3 bucket created
* Bucket policy allowing ALB logging
* ALB in active state

**Step-by-Step Implementation**

**Step 1:** Create S3 Bucket

**Step 2:** Add Logging Policy

**Step 3:** Enable Access Logging

* ALB → Attributes → Enable Logging

**Step 4:** Generate Traffic

Access ALB a few times.

**Step 5:** Verify Logs in S3

* Create a s3 bucket



* S3 bucket has been created successfully

A screenshot of a computer

AI-generated content may be incorrect.

**S3 bucket policy:**

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "AWSALBAccessLogsPolicy",

"Effect": "Allow",

"Principal": {

"Service": "logdelivery.elasticloadbalancing.amazonaws.com"

},

"Action": "s3:PutObject",

"Resource": "arn:aws:s3:::lbs3task/AWSLogs/221427048026/\*"

},

{

"Sid": "AWSALBAccessLogsBucketPermissionsCheck",

"Effect": "Allow",

"Principal": {

"Service": "logdelivery.elasticloadbalancing.amazonaws.com"

},

"Action": [

"s3:GetBucketAcl",

"s3:ListBucket"

],

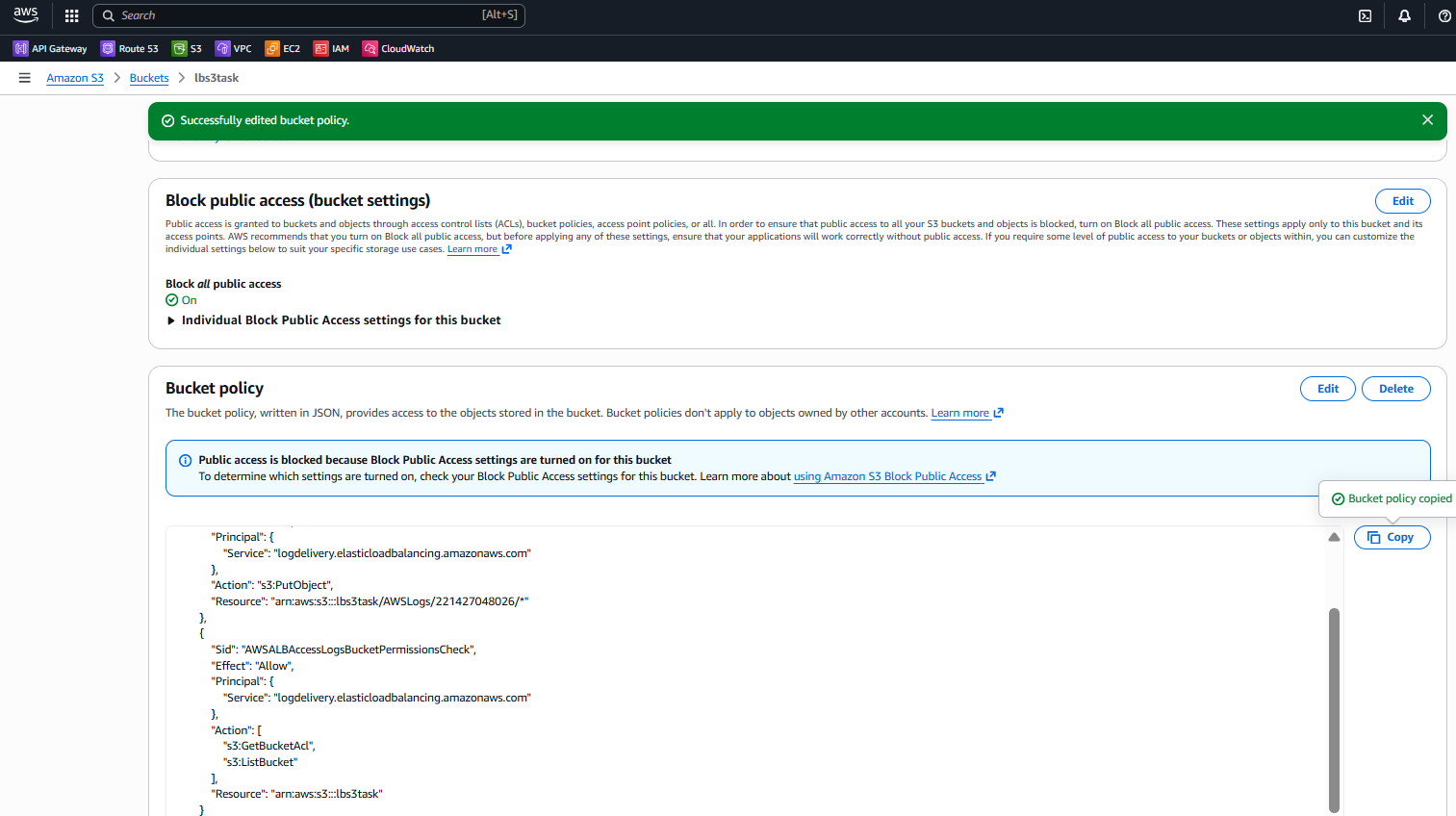
"Resource": "arn:aws:s3:::lbs3task"

}

]

}

* Select s3 bucket that we created > permissions > bucket policy > edit > add above Json format script in it > save policy



* Redirect to load balancer > select load balancer that we created > attributes > click on edit > scroll down to monitoring > select Access logs > browse s3 bucket that we created > click on save edits

A screenshot of a computer

AI-generated content may be incorrect.

* Load balancer attributes have been created successfully

A screenshot of a computer

AI-generated content may be incorrect.

* Redirect to s3 bucket. We can Aws logs here in below screenshot
* Sometimes it takes few minutes to reflect in s3 bucket

A screenshot of a computer

AI-generated content may be incorrect.

* We can check the logs now

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion:**

ALB logs were successfully configured and are being stored in S3 for monitoring and auditing.

**Overall Tasks Conclusion:**

The assignment was successfully completed by implementing multiple AWS load balancing solutions (Classic, Application, and Network Load Balancers) and integrating them with EC2 instances to ensure high availability, scalability, and efficient traffic distribution. SSL/TLS security was applied to the Application Load Balancer to enable secure HTTPS communication, and Route 53 was used for DNS-level traffic routing to provide a user-friendly domain endpoint. Additionally, Application Load Balancer access logs were configured to be pushed to an S3 bucket, enabling centralized logging and improved operational visibility.

Each component was validated thoroughly, ensuring that routing, SSL termination, instance health checks, and S3 logging worked as expected. Overall, the environment now demonstrates a robust, scalable, and secure load balancing architecture aligned with AWS best practices.