**Task on Load Balancers**

**1. Configure Classic Load balancer.**

Classic Load Balancer works at both Layer 4 (TCP) and Layer 7 (HTTP/HTTPS) of the OSI model and is suited for EC2 instances in one or more Availability Zones.

**Step 1 — Log in to AWS Console**

1. Go to the **AWS Management Console**.
2. Search for **EC2** service and click it.

**Step 2 — Navigate to Load Balancers**

1. In the EC2 dashboard, scroll to **Load Balancing** → **Load Balancers**.
2. Click **Create Load Balancer**.
3. Select **Classic Load Balancer** and click **Create**.

**Step 3 — Configure Basic Settings**

1. **Name**: Give your load balancer a name (e.g., My-CLB).
2. **Create LB inside a VPC**: Select your VPC.
3. **Scheme**:
   * Internet-facing → for public access.
   * Internal → for internal access.
4. **Listeners**:  
   Default is HTTP:80. Add HTTPS:443 if needed.

**Step 4 — Configure Security Settings (If HTTPS)**

If you added HTTPS:

1. Select an **SSL Certificate** (from ACM or upload your own).
2. Choose the **security policy**.

**Step 5 — Configure Security Groups**

Attach a security group that allows traffic on the listener ports (HTTP 80, HTTPS 443).

**Step 6 — Configure Health Check**

Classic LB requires a health check configuration.

1. **Ping Protocol**: HTTP or TCP.
2. **Ping Path**: / (for HTTP; can be your app health endpoint, e.g., /health).
3. **Response Timeout**: 5 seconds (default).
4. **Interval**: 30 seconds (default).
5. **Unhealthy Threshold**: 2 (default).
6. **Healthy Threshold**: 2 (default).

**Step 7 — Add EC2 Instances**

Select one or more EC2 instances you want to attach to the CLB.  
These should be running in the same VPC and accessible via listener ports.

**Step 8 — Add Tags**

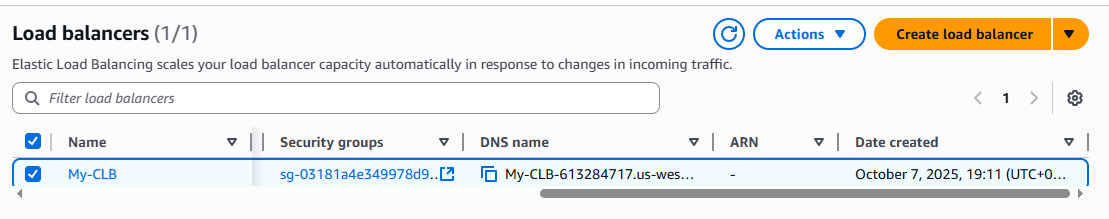
(Optional) Add tags for better organization.

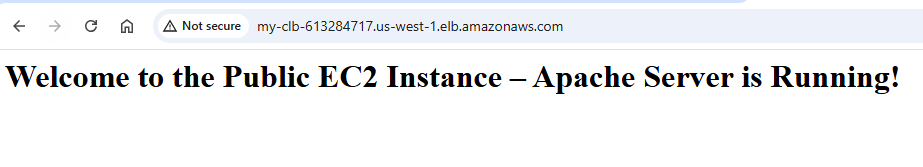
**Step 9 — Review & Create**

Review all settings → Click **Create**.

**Step 10 — Test**

1. Go to the **Load Balancers** dashboard.
2. Copy the **DNS name** of your CLB.
3. Open the DNS name in a browser (for HTTP/HTTPS) to verify it routes traffic correctly.





**2. Configure Application Load balancer.**

### **Step 1 — Login to AWS Console**

1. Go to **AWS Management Console**.
2. Search for **EC2** and open it.
3. In the left menu, scroll to **Load Balancing** → **Load Balancers**.

### **Step 2 — Create an Application Load Balancer**

1. Click **Create Load Balancer**.
2. Select **Application Load Balancer** → Click **Create**.

### **Step 3 — Configure Basic Settings**

* **Name**: Give a unique name (e.g., My-ALB).
* **Scheme**:
  + Internet-facing → for public access.
  + Internal → for private access.
* **IP address type**: IPv4 or Dualstack.
* **Listeners**: Default is HTTP (80). You can add HTTPS (443).

### **Step 4 — Configure Availability Zones**

* Select your **VPC**.
* Select **Availability Zones** where your EC2 instances are running.
* Select the **subnets** (at least two for high availability).

### **Step 5 — Configure Security Groups**

* Attach a **security group** that allows inbound traffic on listener ports (80 for HTTP, 443 for HTTPS).
* Also allow traffic from ALB to EC2 instances.

### **Step 6 — Configure Routing (Target Group)**

ALB uses target groups to route requests.

1. **Target Group Name**: e.g., MyTargetGroup.
2. **Target Type**:
   * Instance → Direct EC2 instances.
   * IP → IP-based routing.
   * Lambda function → Serverless.
3. **Protocol**: HTTP or HTTPS.
4. **Port**: e.g., 80.
5. **Health Check Path**: / or /health.
6. Click **Next**.

### **Step 7 — Register Targets**

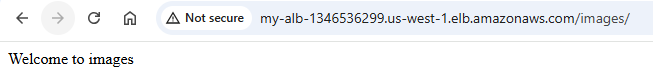
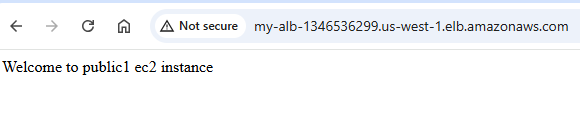
* Select the EC2 instances you want ALB to route traffic to.
* Click **Add to registered**.
* Click **Next**.

### **Step 8 — Review & Create**

* Review your configuration.
* Click **Create Load Balancer**.

### **Step 9 — Test Your ALB**

1. In the **Load Balancers** list, select your ALB.
2. Copy its **DNS name**.
3. Open the DNS name in a browser to verify routing works.



**4 .Attach SSL for application load balancer.**

**5.Map Application load balancer to R53.**

### **Get an SSL Certificate (ACM)**

#### **Option A: Using Your Own Domain (e.g.** sravandevops.co.in**)**

1. Go to **AWS Console → AWS Certificate Manager (ACM)**
2. Click **“Request a certificate”**
3. Choose **Request a public certificate** → click **Next**
4. Add your domain name(s):
   * Example:
   * sravandevops.co.in
   * \*.sravandevops.co.in
   * The \*. covers subdomains (like app.sravandevops.co.in)
5. Choose **DNS validation** (recommended)
6. Click **Request**

### **2️⃣ Validate the Certificate**

1. After requesting, you’ll see a **CNAME record** ACM gives you.
2. Go to **Route 53 → Hosted Zone → your domain**
3. Add that **CNAME record** exactly as shown in ACM.
4. Within a few minutes, the certificate status changes from **“Pending validation” → “Issued”**

✅ Once it says **Issued**, your SSL is ready.

### **3️⃣ Edit ALB Listener to Add HTTPS (443)**

1. Go to **EC2 → Load Balancers**
2. Select your **Application Load Balancer**
3. Go to the **Listeners** tab
4. Click **“Add listener”**
   * **Protocol:** HTTPS
   * **Port:** 443
5. Under **Default action**, choose:
   * **Forward to:** your target group (example: Home)

### **4️⃣ Attach the SSL Certificate**

In the same listener creation window:

1. Under **Default SSL certificate**, choose:
   * **From ACM (recommended)**
   * Then select your validated certificate (sravandevops.co.in)
2. (Optional) Choose **Security Policy** — e.g. ELBSecurityPolicy-2016-08 (default, safe option)

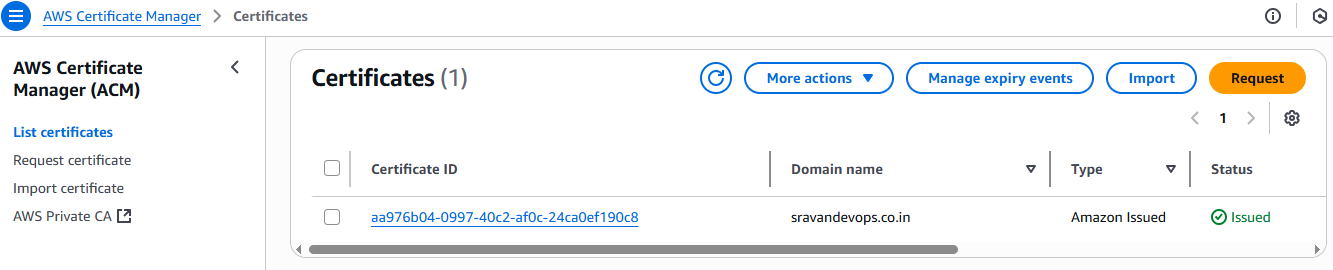
Click **Add listener**.

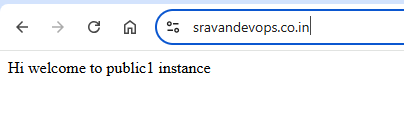
### **5️⃣ Redirect HTTP → HTTPS (optional but recommended)**

You can make sure all traffic uses HTTPS automatically.

1. Under your **HTTP:80 listener** → click **View/edit rules**
2. Add a rule:
   * **IF:** Path is / (or leave default)
   * **THEN:** Action → **Redirect to → HTTPS (port 443)**
   * Choose “Permanent (301)” redirect

Now all HTTP requests will automatically go to HTTPS.





**5.Map Application load balancer to R53.**

**Step 1 — Create an S3 Bucket for ALB Logs**

1. Go to the **AWS Management Console → S3**.
2. Click **Create bucket**.
3. Fill in:
   * **Bucket name:** alb-logs-myapp
   * **Region:** N. California (us-west-1)
4. Uncheck **Block all public access** (optional — logging data isn’t public anyway).
5. Click **Create bucket**.

**Step 2 — Enable ALB Access Logging**

1. Go to **EC2 → Load Balancers** in AWS Console.
2. Select your **Application Load Balancer** (MyVPC-ALB).
3. Go to the **Attributes** tab → **Edit attributes**.
4. Turn **Access logs** **ON**.

Fill in:

* **S3 location:** select your bucket alb-logs-myapp.
* **Prefix:** (optional) something like alb-logs/ — this organizes logs inside the bucket.
* **Enable:** ✓

1. Click **Save changes**.

**Step 3 — Grant ALB Permission to Write Logs to S3**

ALB needs permission to write logs.  
AWS will normally prompt you to add the correct **bucket policy**.

If not, add this policy to your bucket:

{

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

"Statement": [

{

"Effect": "Allow",

"Principal": {

"AWS": "arn:aws:iam::027434742980:root"

},

"Action": "s3:PutObject",

"Resource": "arn:aws:s3:::applicationlb-logs-s3/AWSLogs/342374577013/\*"

}

]

}

**Step 4 — Verify Logs**

After enabling logging, ALB will start storing logs every ~5 minutes.

Go to:

S3 → alb-logs-myapp → alb-logs/

You’ll see log files like:

AWSLogs/<AWS\_ACCOUNT\_ID>/elasticloadbalancing/us-west-1/YYYY/MM/DD/<log-file>.log.gz

