**Auto scaling Tasks**

1. **Create one VPC in N. Virginia region.**

**Open the VPC Dashboard**

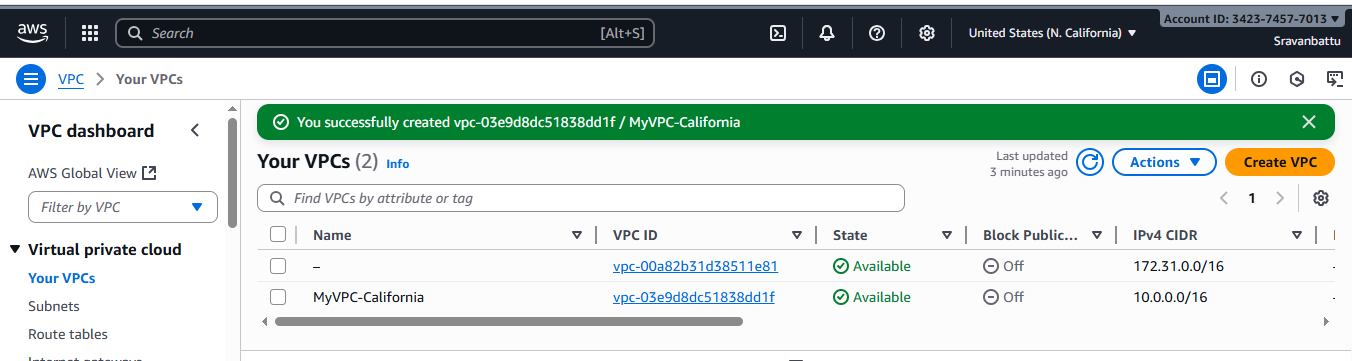
1. In the **search bar**, type **VPC**.
2. Click on **VPC** to open the dashboard.

**Create a New VPC**

1. Click on **“Create VPC”**.
2. Choose **“VPC only”** under *Resources to create*.
3. Fill in the details:
   * **Name tag:** MyVPC-California
   * **IPv4 CIDR block:** 10.0.0.0/16
   * **IPv6 CIDR block:** *No IPv6 CIDR block*
   * **Tenancy:** Default
4. Click **Create VPC**.

**Verify the VPC**

1. Once created, you’ll see a success message.
2. Go to **Your VPCs** in the left menu.
3. Confirm that your new VPC (e.g., MyVPC-California) is listed with:
   * **VPC ID**
   * **CIDR Block:** 10.0.0.0/16
   * **Region:** N. California (us-west-1)



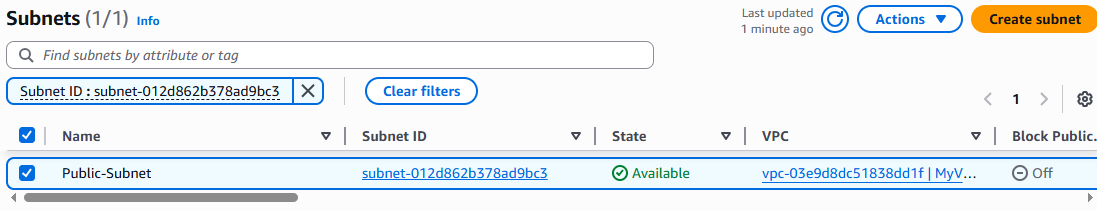
**2.Create two subnets: one public subnet and one private subnet.**

## Step 1: Open the VPC Dashboard

1. Go to the **AWS Management Console** → Search for **VPC** → open it.
2. Make sure the **region** is set to **N. California (us-west-1)**.

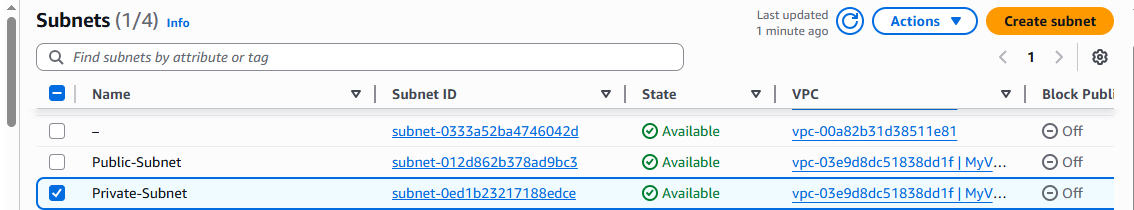
## 🌐 Step 2: Create the Public Subnet

1. In the left menu, click **Subnets** → **Create subnet**.
2. Under **VPC ID**, select your previously created VPC (e.g., MyVPC-California).
3. Add subnet details:
   * **Subnet name:** Public-Subnet
   * **Availability Zone:** us-west-1a
   * **IPv4 CIDR block:** 10.0.1.0/24
4. Click **Create subnet**.
5. After creation, select the subnet → choose **Actions → Edit subnet settings** →  
   Enable **Auto-assign public IPv4 address** → **Save**.



## Step 3: Create the Private Subnet

1. Again, click **Create subnet**.
2. Choose the same **VPC (MyVPC-California)**.
3. Add subnet details:
   * **Subnet name:** Private-Subnet
   * **Availability Zone:** us-west-1b
   * **IPv4 CIDR block:** 10.0.2.0/24
4. Click **Create subnet**.



**3.Attach an IGW to the VPC.**

## Step 1: Open the VPC Dashboard

1. In the AWS Console, search for **VPC** and open it.
2. Confirm your region is **N. California (us-west-1)**.

## 🛠️ Step 2: Create an Internet Gateway

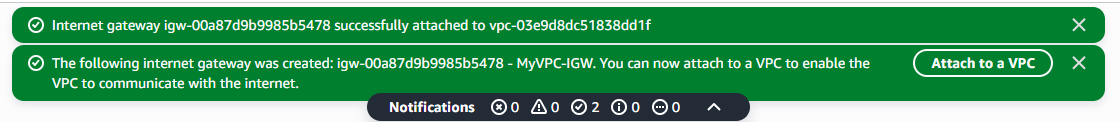
1. In the left menu, click **Internet Gateways** → **Create internet gateway**.
2. Fill in:
   * **Name tag:** MyVPC-IGW
3. Click **Create internet gateway**.

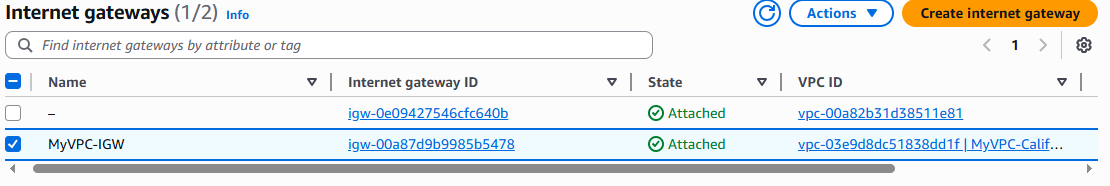
## 🔗 Step 3: Attach IGW to Your VPC

1. Select the newly created IGW (MyVPC-IGW).
2. Click **Actions → Attach to VPC**.
3. Choose your **VPC (MyVPC-California)** from the list.
4. Click **Attach internet gateway**.

## ✅ Step 4: Verify

* The **State** of the Internet Gateway should now show **“Attached”**.
* You can also see the **VPC ID** column displaying your VPC.





**4.Create one public route table (RT) and one private route table.**

## Step 1: Open the VPC Dashboard

1. In the AWS Console, go to **VPC → Route Tables**.
2. Make sure your region is **N. California (us-west-1)**.

## 🌐 Step 2: Create the Public Route Table

1. Click **Create route table**.
2. Enter:
   * **Name tag:** Public-RT
   * **VPC:** Select your VPC (MyVPC-California)
3. Click **Create route table**.

## 🔗 Step 3: Add Route to Internet Gateway

1. Select the **Public-RT** → go to the **Routes** tab → click **Edit routes** → **Add route**.
2. Fill in:
   * **Destination:** 0.0.0.0/0
   * **Target:** Select your **Internet Gateway (MyVPC-IGW)**.
3. Click **Save changes**.

✅ This gives internet access to anything in subnets associated with this route table.

## 🗺️ Step 4: Associate Public Subnet

1. With **Public-RT** selected, go to **Subnet associations → Edit subnet associations**.
2. Select your **Public-Subnet (10.0.1.0/24)**.
3. Click **Save associations**.

## 🔒 Step 5: Create the Private Route Table

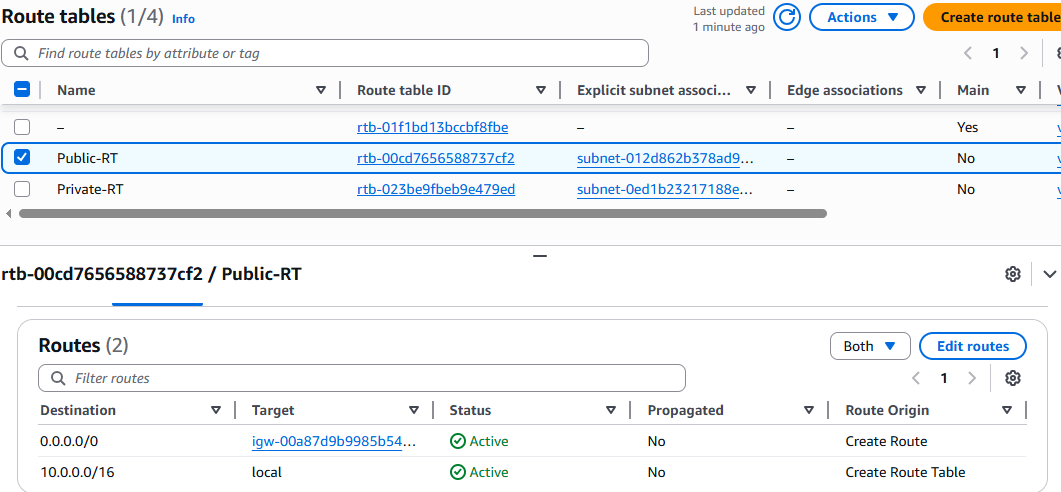
1. Click **Create route table** again.
2. Enter:
   * **Name tag:** Private-RT
   * **VPC:** Select your **MyVPC-California**.
3. Click **Create route table**.

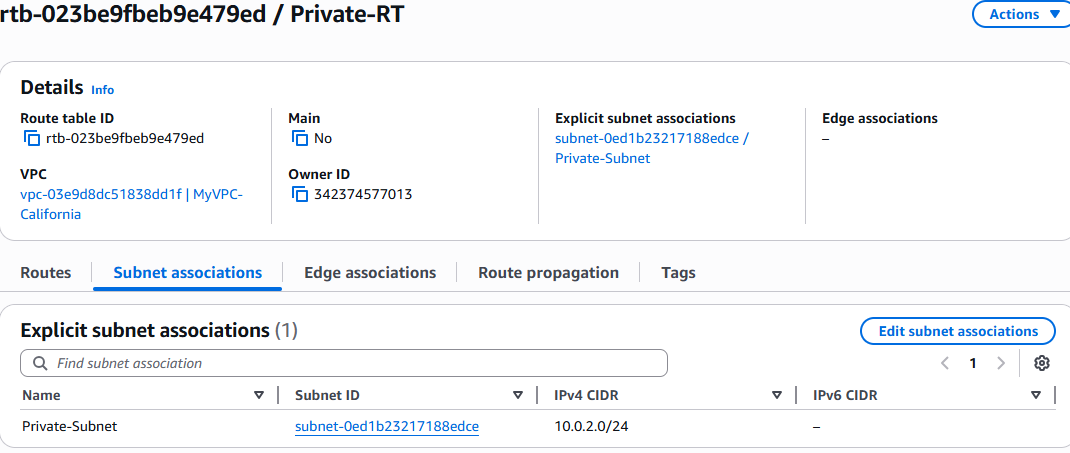
## 🚫 Step 6: Keep Private RT Internal

* Don’t add any route to the Internet Gateway.
* The default local route (10.0.0.0/16 → local) allows communication within the VPC only.

## 🧩 Step 7: Associate Private Subnet

1. Select **Private-RT** → **Subnet associations → Edit subnet associations**.
2. Select your **Private-Subnet (10.0.2.0/24)**.
3. Click **Save associations**.





**5.Deploy a NAT gateway in the public subnet and attach the NAT gateway to the private subnet.**

## Step 1: Open the VPC Dashboard

1. Go to the **AWS Management Console → VPC service**.
2. Confirm your region is **N. California (us-west-1)**.

## ⚙️ Step 2: Create an Elastic IP (EIP)

The NAT Gateway needs a public Elastic IP to route traffic.

1. In the left menu, click **Elastic IPs** → **Allocate Elastic IP address**.
2. Choose:
   * **Network border group:** Keep the default (us-west-1)
3. Click **Allocate**.
4. Note the **Elastic IP Allocation ID** (you’ll need it next).

## 🛠️ Step 3: Create the NAT Gateway

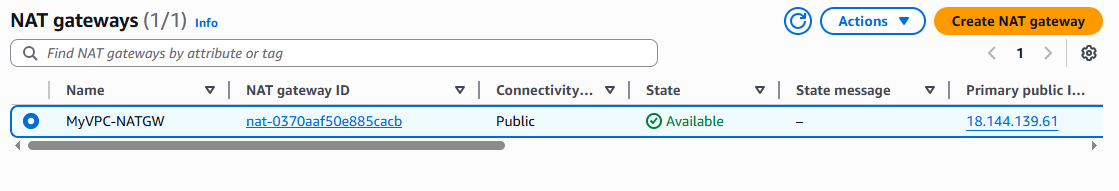
1. In the left menu, click **NAT Gateways → Create NAT gateway**.
2. Fill in:
   * **Name:** MyVPC-NATGW
   * **Subnet:** Choose your **Public-Subnet (10.0.1.0/24)**
   * **Elastic IP allocation ID:** Select the one you just created.
3. Click **Create NAT gateway**.

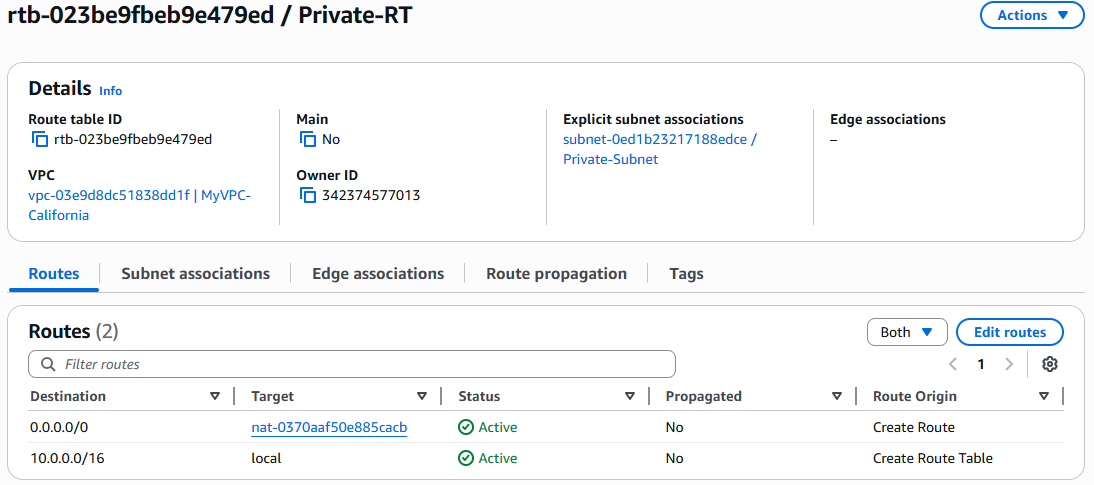
✅ Wait until the **Status** changes from Pending → Available (may take a minute).

## 🗺️ Step 4: Update the Private Route Table

Now route all outbound internet traffic from the **private subnet** through the **NAT Gateway**.

1. Go to **Route Tables** in the left menu.
2. Select your **Private-RT**.
3. Under the **Routes** tab → click **Edit routes → Add route**.
4. Fill in:
   * **Destination:** 0.0.0.0/0
   * **Target:** Choose **NAT Gateway → MyVPC-NATGW**
5. Click **Save changes**.





**6.Create two instances, one in the public subnet and one in the private subnet.**

## Step 1: Open EC2 Dashboard

1. In the **AWS Console**, search for **EC2** and open it.
2. Confirm the region: **N. California (us-west-1)**

## ☁️ Step 2: Create a Key Pair (if not done already)

1. In the EC2 dashboard, click **Key pairs → Create key pair**.
2. Enter:
   * **Name:** MyVPC-Key
   * **Key pair type:** RSA
   * **File format:** .pem (for Linux/Mac) or .ppk (for Windows PuTTY)
3. Click **Create key pair** — it’ll auto-download.

You’ll need this to SSH into your EC2 instances later.

## 🖥️ Step 3: Launch the Public Instance

1. Click **Instances → Launch instances**.
2. Configure:
   * **Name:** Public-Instance
   * **AMI:** Amazon Linux 2023 (Free tier eligible)
   * **Instance type:** t2.micro
   * **Key pair:** MyVPC-Key
3. Under **Network settings**:
   * **VPC:** MyVPC-California
   * **Subnet:** Public-Subnet (10.0.1.0/24)
   * **Auto-assign public IP:** **Enable**
4. Under **Security group:**
   * Create a new SG named Public-SG
   * Add inbound rules:
     + **SSH (22)** → Source: My IP
     + **HTTP (80)** → Source: 0.0.0.0/0 (optional)
5. Click **Launch instance**.

✅ The public instance will have a **public IP address** — note it down for SSH access.

## 🔒 Step 4: Launch the Private Instance

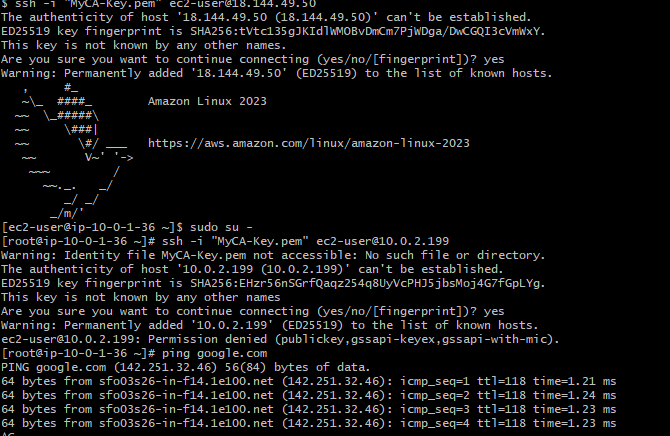
1. Again, click **Launch instance**.
2. Configure:
   * **Name:** Private-Instance
   * **AMI:** Amazon Linux 2023
   * **Instance type:** t2.micro
   * **Key pair:** MyVPC-Key
3. Under **Network settings**:
   * **VPC:** MyVPC-California
   * **Subnet:** Private-Subnet (10.0.2.0/24)
   * **Auto-assign public IP:** **Disable**
4. Under **Security group:**
   * Create a new SG named Private-SG
   * Add inbound rule:
     + **SSH (22)** → Source: Public-SG (so only the public instance can access it)
5. Click **Launch instance**.

✅ The private instance will **not** have a public IP — meaning it’s isolated from the internet directly, but can access it via the NAT Gateway.

## 🧩 Step 5: Test Connectivity

1. From your local terminal, SSH into the **public instance**:
2. ssh -i MyVPC-Key.pem ec2-user@<Public-Instance-Public-IP>
3. From inside the public instance, SSH into the **private instance**:
4. ssh ec2-user@<Private-Instance-Private-IP>
5. Test internet connectivity from the private instance:

ping google.com



**7.Deploy Apache server on both EC2 instances with a sample index.html file.**

## Step 1: Connect to Your EC2 Instances

### 🖥️ Public Instance

From your **local machine**, SSH into your **public EC2**:

ssh -i MyVPC-Key.pem ec2-user@<Public-Instance-Public-IP>

### 🔒 Private Instance

From inside your **public EC2**, SSH into your **private EC2**:

ssh ec2-user@<Private-Instance-Private-IP>

## ⚙️ Step 2: Install Apache (on Both Instances)

Run the following commands **on each EC2 instance** (one by one):

sudo yum update -y

sudo yum install httpd -y

sudo systemctl enable httpd

sudo systemctl start httpd

✅ This installs Apache, enables it at boot, and starts the service.

## 🧾 Step 3: Create a Sample Web Page

Run this on each instance, customizing the message slightly so you can tell them apart:

### On the ****Public Instance****

echo "<h1>Welcome to the Public EC2 Instance – Apache Server is Running!</h1>" | sudo tee /var/www/html/index.html

### On the ****Private Instance****

echo "<h1>Welcome to the Private EC2 Instance – Apache Server is Running!</h1>" | sudo tee /var/www/html/index.html

## 🔥 Step 4: Allow HTTP (Port 80) Access

### For the Public Instance:

1. Go to **EC2 → Security Groups → Public-SG**.
2. Add an inbound rule:
   * **Type:** HTTP
   * **Port:** 80
   * **Source:** 0.0.0.0/0

### For the Private Instance:

You don’t need external HTTP access, but if you want to test it internally:

1. Go to **Security Group → Private-SG**.
2. Add inbound rule:
   * **Type:** HTTP
   * **Port:** 80
   * **Source:** Public-SG

## 🧩 Step 5: Test Your Setup

### ✅ Public Instance:

From your browser, open:

http://<Public-Instance-Public-IP>

You should see:

**Welcome to the Public EC2 Instance – Apache Server is Running!**

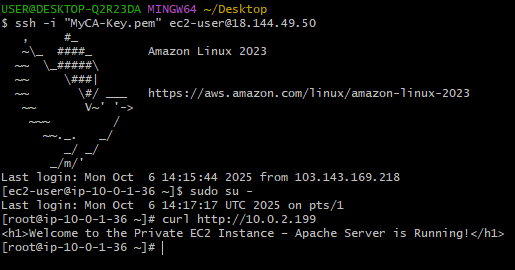
### ✅ Private Instance:

From inside the **public EC2**, test with:

curl http://<Private-Instance-Private-IP>

You should see:

**Welcome to the Private EC2 Instance – Apache Server is Running!**



**8.Create one application load balancer and attach it to both EC2 instances.**

Perfect 😄 — let’s create an **Application Load Balancer (ALB)** in your **N. California (us-west-1)** VPC and attach both EC2 instances (public + private).

This will allow traffic to be balanced between them.

## **Step 1 — Open EC2 → Load Balancers**

1. Go to **AWS Management Console** → search for **EC2** → open **Load Balancers** (under Load Balancing).
2. Click **Create Load Balancer**.
3. Choose **Application Load Balancer** → click **Create**.

### ****Step 2 — Configure Load Balancer****

* **Name:** MyVPC-ALB
* **Scheme:** Internet-facing (so it can accept traffic from the internet)
* **IP address type:** IPv4
* **Listeners:** Keep default HTTP:80 (can add HTTPS later)
* **Availability Zones:**
  + Select your **VPC (MyVPC-California)**
  + Select both subnets:
    - Public Subnet (10.0.1.0/24)
    - Private Subnet (10.0.2.0/24) — ALBs require at least two AZs, so this works fine.

Click **Next**.

### ****Step 3 — Configure Security Groups****

1. Create or select a security group for ALB:
   * **Name:** ALB-SG
   * Allow inbound HTTP (port 80) from 0.0.0.0/0
   * Allow inbound HTTPS (port 443) if you plan SSL later
   * Outbound: all traffic

Click **Next**.

### ****Step 4 — Configure Target Group****

1. **Target group name:** MyVPC-TargetGroup
2. **Target type:** Instance
3. **Protocol:** HTTP
4. **Port:** 80
5. **VPC:** MyVPC-California

Click **Next**.

### ****Step 5 — Register Targets (EC2 Instances)****

1. Select both your instances:
   * **Public EC2**
   * **Private EC2**
2. Click **Include as pending below** → **Create target group**.

### ****Step 6 — Review and Create****

Review all settings → click **Create Load Balancer**.

AWS will take a few seconds to set up your ALB.

## **Step 7 — Verify**

1. Go to your **ALB → Description** → note the **DNS name** (something like:

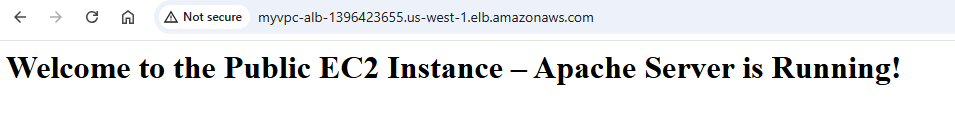
myvpc-alb-1234567890.us-west-1.elb.amazonaws.com

```)

2. Open it in your browser:

http://myvpc-alb-1234567890.us-west-1.elb.amazonaws.com

You should see your `index.html` served from either public or private EC2 instance — traffic will be





**9.Store application load balancer logs in S3.**

Perfect 👍 — storing **Application Load Balancer (ALB) logs** in an S3 bucket is a best practice so you can later analyze traffic and debug issues.

Here’s the step-by-step guide for **N. California (us-west-1)**:

**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**

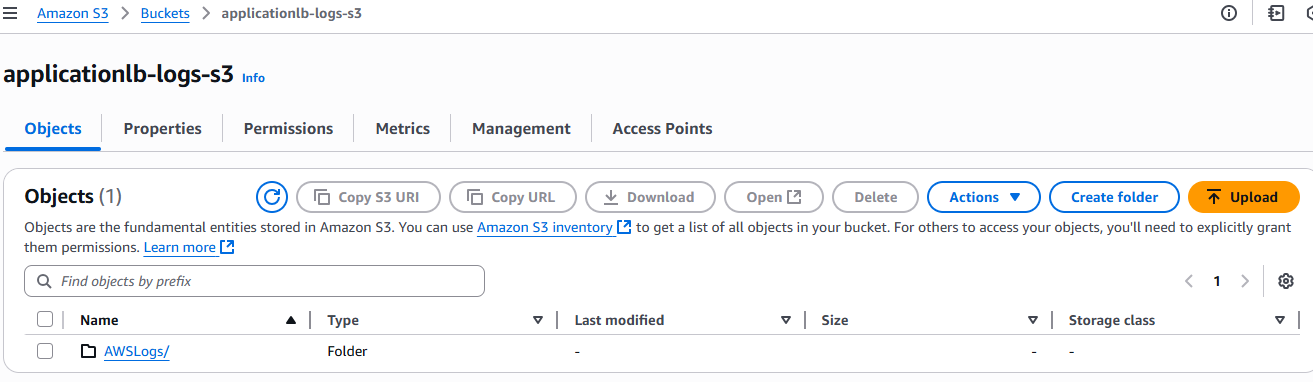
After enabling logging, ALB will start storing logs every ~5 minutes.

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



**10.Store the VPC flow logs in a CloudWatch log group.**

## **Step-by-Step: Store VPC Flow Logs in CloudWatch Logs**

### ****Step 1 — Open your VPC****

1. Go to the **AWS Management Console → VPC service**.
2. On the left panel, select **Your VPCs**.
3. Select your **VPC** (the one you created in N. California).

### ****Step 2 — Create a Flow Log****

1. With your VPC selected, click the **Flow logs** tab.
2. Click **Create flow log**.

### ****Step 3 — Configure the Flow Log****

**Step 4 — Create IAM Role (if not existing)**

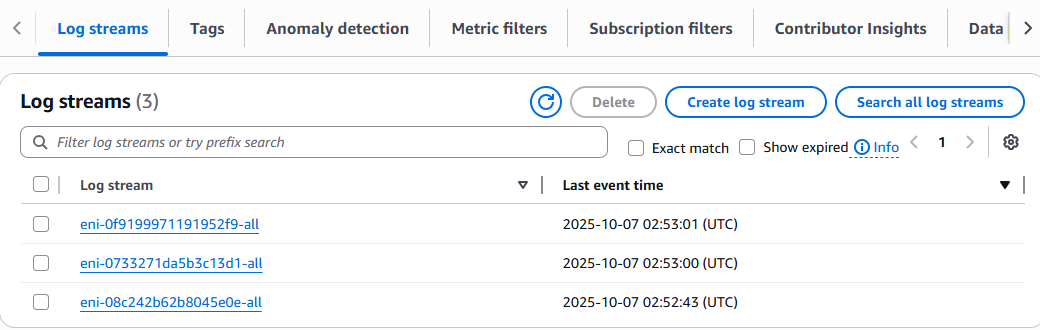
### ****Create the Flow Log****

Click **Create flow log**

### ****Verify in CloudWatch****

1. Go to **CloudWatch → Logs → Log groups**.
2. You should see your new group:

/aws/vpc/flow-logs/VPCFlowLogsGroup



**11.Create monitoring dashboards to monitor CPU utilization and to monitor the Apache service.**

## Step 1: Install CloudWatch Agent (Amazon Linux 2023)

Run these commands:

sudo dnf install -y amazon-cloudwatch-agent

