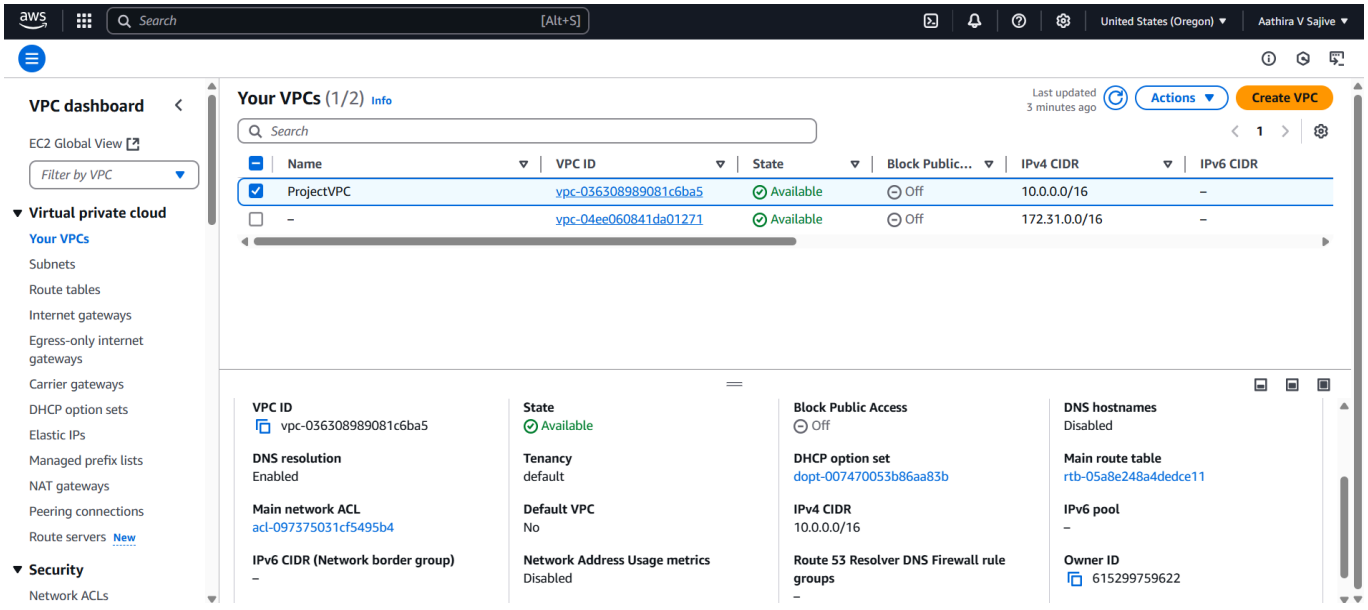
Auto Scaling Web Infrastructure with Elastic Load Balancer

Objective:

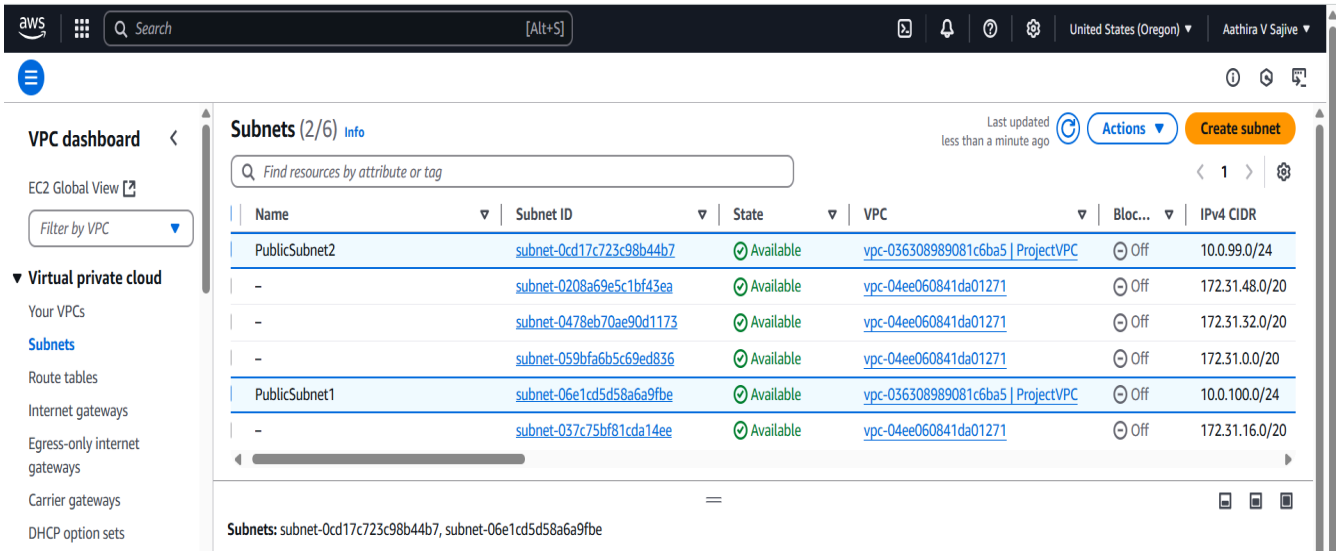
To deploy a scalable and highly available web server infrastructure by creating a custom VPC with two public subnets across different Availability Zones, launching two EC2 instances (one in each subnet), and configuring an Application Load Balancer (ELB) within the same VPC. The setup ensures high availability and fault tolerance, with Auto Scaling triggered based on CPU utilization, using the same Load Balancer for traffic distribution and instance health monitoring.

Launch VPC:



- **Log in to AWS Console**
- **Choose theRegion**
- **Navigate to VPC Dashboard > Your VPCs > Create VPC**
- **Choose:**
    - **Name tag: ProjectVPC**
    - **IPv4 CIDR block: 10.0.0.0/16**
- **Click Create VPC**

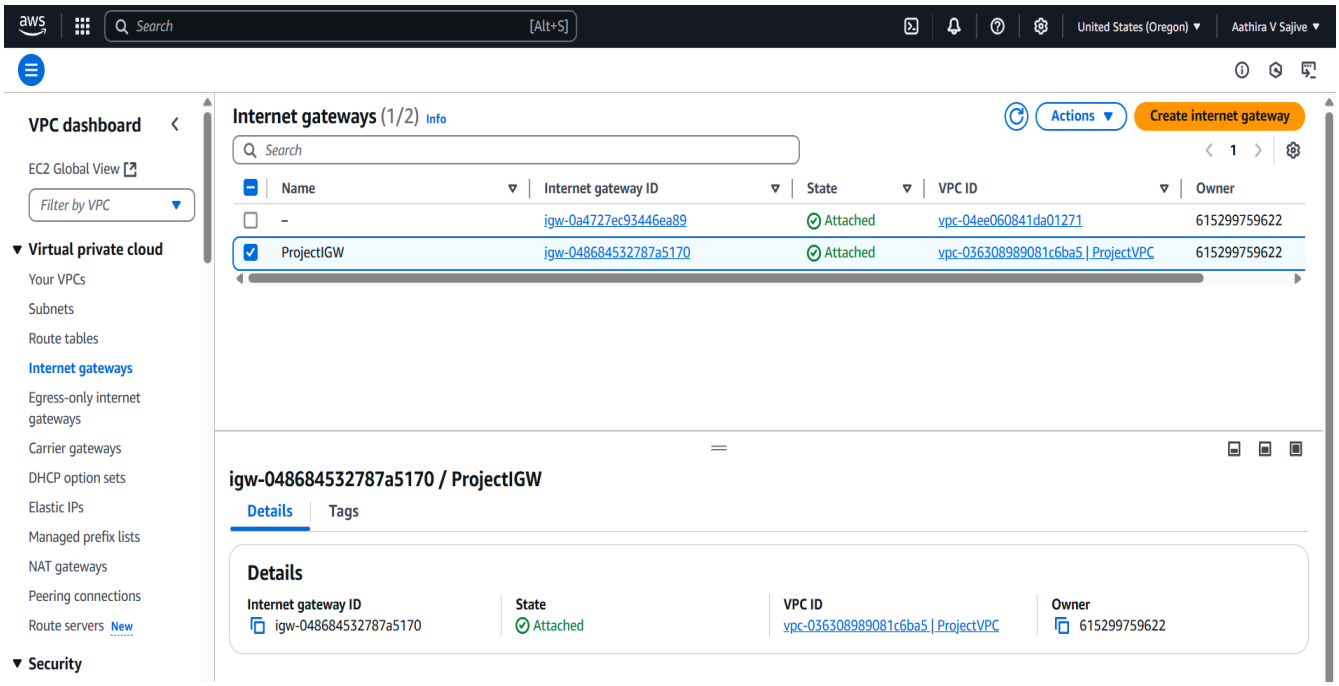
Create Subnets



- **Go to VPC Dashboard > Subnets > Create subnet**
- **Select your ProjectVPC**
- **Name:**
    - **PublicSubnet1**
    - **PublicSubnet2**
- **Availability Zones:**
    - **Choose different AZs: us-west-2a and us-west-2b**

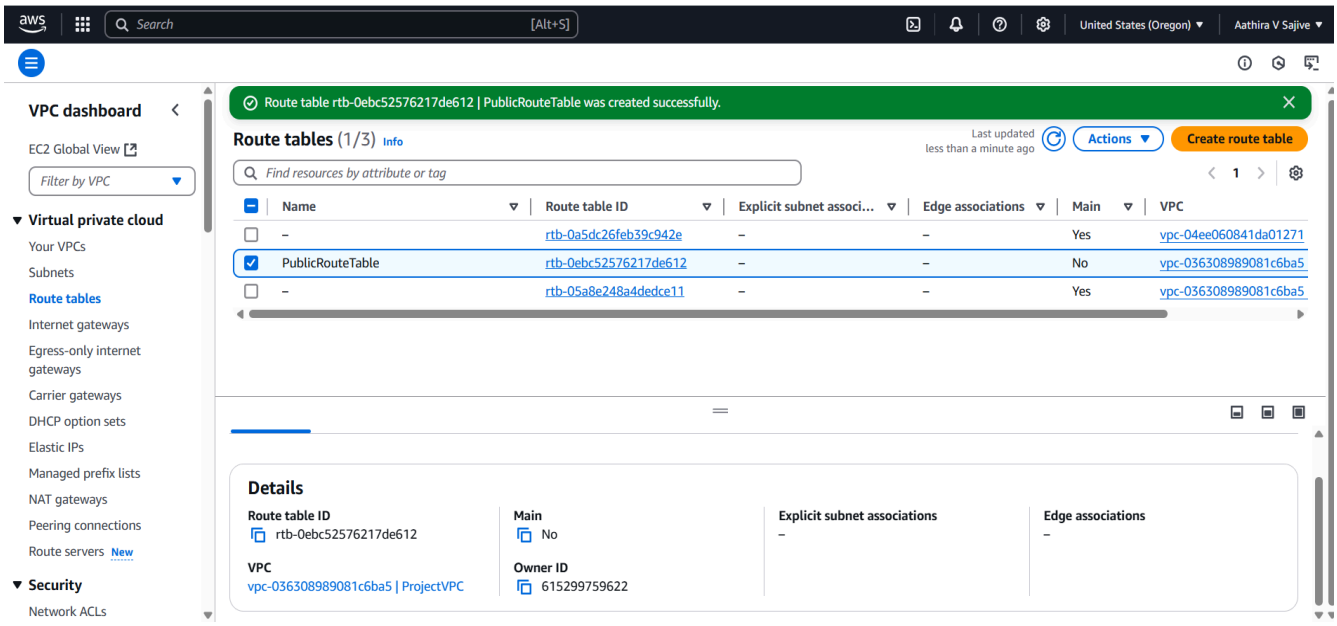
- **IPv4 CIDR blocks:**
  - **10.0.100.0/24 → for Public Subnet 1**
  - **10.0.99.0/24 → for Public Subnet 2**
- **Launch both Subnets**

Create Internet Gateway



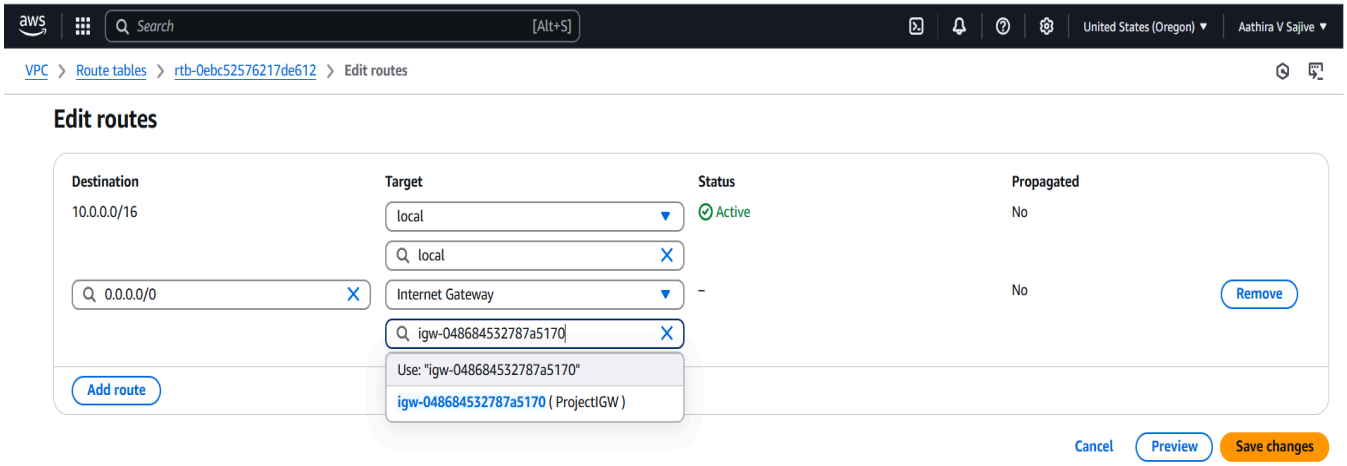
- **Go to VPC Dashboard > Internet Gateways > Create internet gateway**
- **Name: ProjectIGW**
- **Click Create Internet Gateway**
- **Select ProjectIGW, then choose Actions > Attach to VPC and attach it to your ProjectVPC.**

Create Route Tables



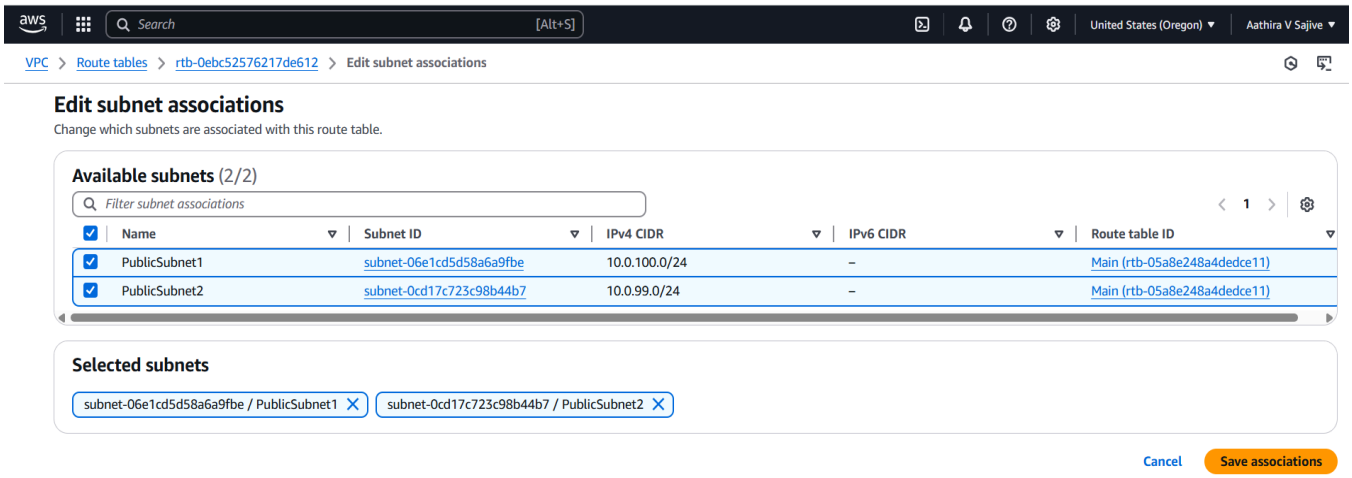
- Go to VPC Dashboard > Route Tables > Create route table
- Name tag: PublicRouteTable
- VPC: Choose your VPC
- Click Create route table

Add Route to the Internet Gateway



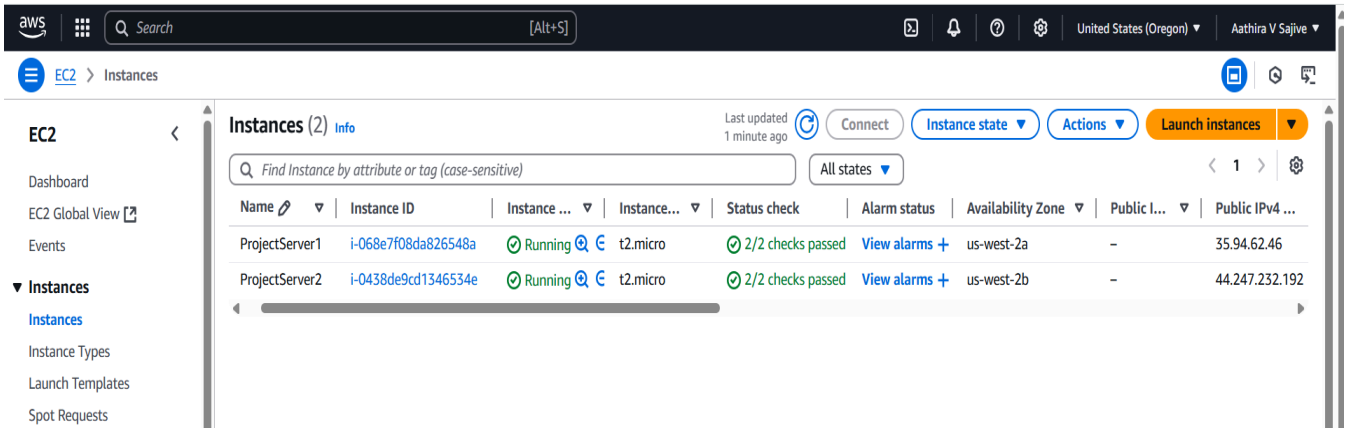
- Select the PublicRouteTable
- Go to the "Routes" tab > click "Edit routes"
- Click Add route:
    - Destination: 0.0.0.0/0
    - Target: Select your ProjectIGW
- Click Save changes

Associate Both Public Subnets



- In the same RouteTable > Go to the "Subnet associations" tab
- Click Edit subnet associations
- Select both:
    - PublicSubnet1 → 10.0.100.0/24
    - PublicSubnet2 → 10.0.99.0/24
- Save

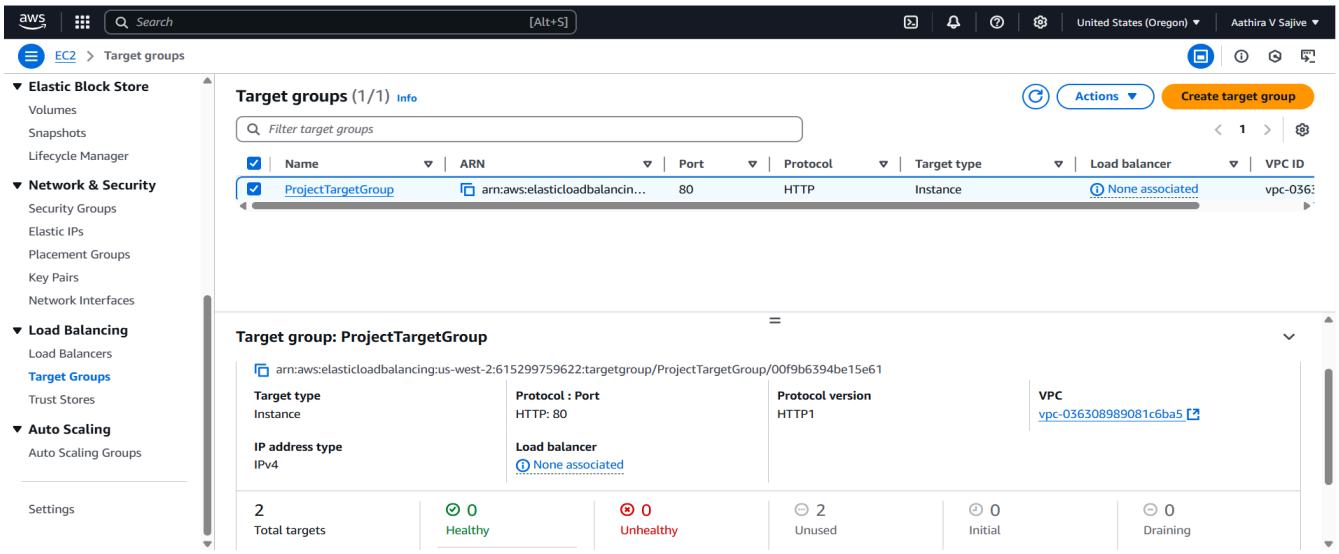
Launch EC2 Instance



- **Go to EC2 Dashboard > Instances > Launch Instance**
- **Name:**
  - **ProjectServer1**
  - **ProjectServer2**
- **AMI: Choose Amazon Linux AMI**
- **Instance type: t2.micro**
- **Key pair: Select or create a new key**
- **Network Settings:**
  - **VPC: your ProjectVPC**
  - **Subnet:**
    - **PublicSubnet1**
    - **PublicSubnet2**
  - **Auto-assign public IP: Enabled**
- **Security Group:**
  - **Create or select a group that allows:**
    - **SSH (port 22)**
    - **HTTP (port 80)**
- **Add User Data Script in: Advanced > User data field:**
- **Launch both Instances**

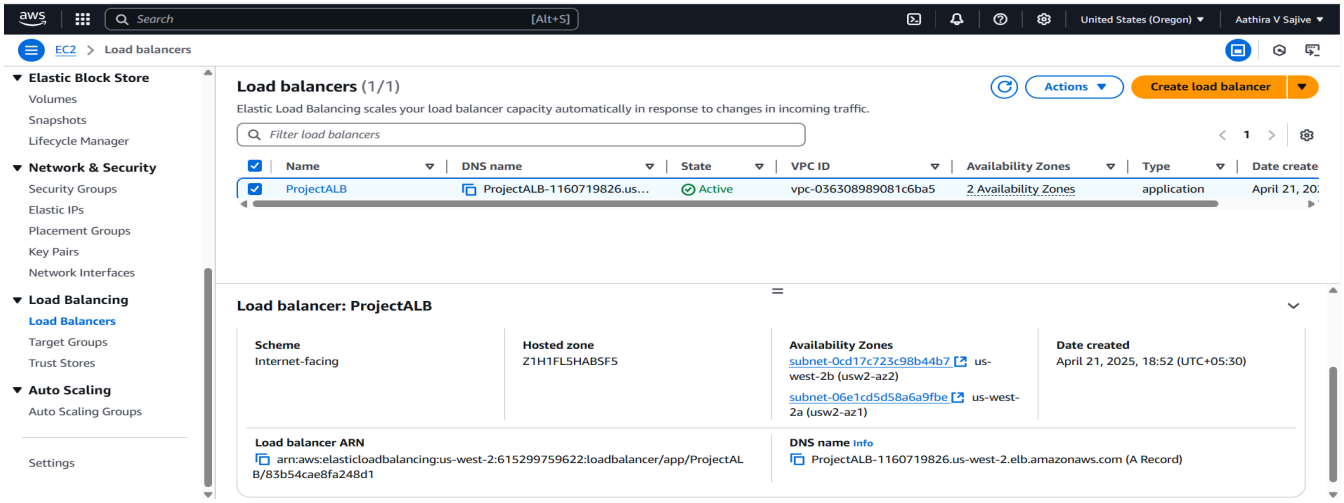
Configure Elastic Load Balancer (ELB)

Create a Target Group



- **Go to EC2 Dashboard > Target Group > Create a new target group**
- **Target type: Instance**
- **Target group name: ProjectTargetGroup**
- **Protocol: HTTP**
- **Port: 80**
- **VPC:Your ProjectVPC**
- **Create a target group.**
- **Register your both EC2 instances into the target group**

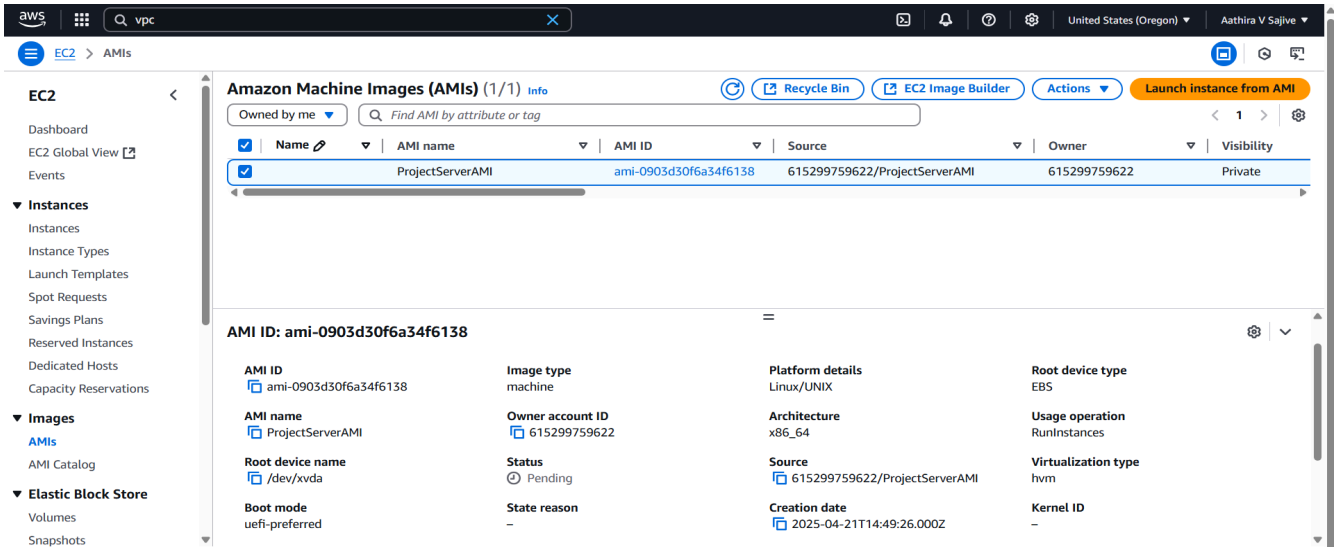
Create an Application Load Balancer



- **Go to EC2 Dashboard > Load Balancers**
- **Click Create Load Balancer > Choose Application Load Balancer**
- **Set:**
  - **Name: ProjectALB**
- **Network mapping:**
  - **VPC: your ProjectVPC**
  - **Availability Zones:**
    - **Choose different AZs: us-west-2a and us-west-2b and attach:**
      - **PublicSubnet1**
      - **PublicSubnet2**
- **Security Group:**
  - **Create or choose one that allows: HTTP (port 80)**
- **Listeners:**
  - **Listener on port 80 → forward to your ProjectTargetGroup**
- **Create Load Balancer**

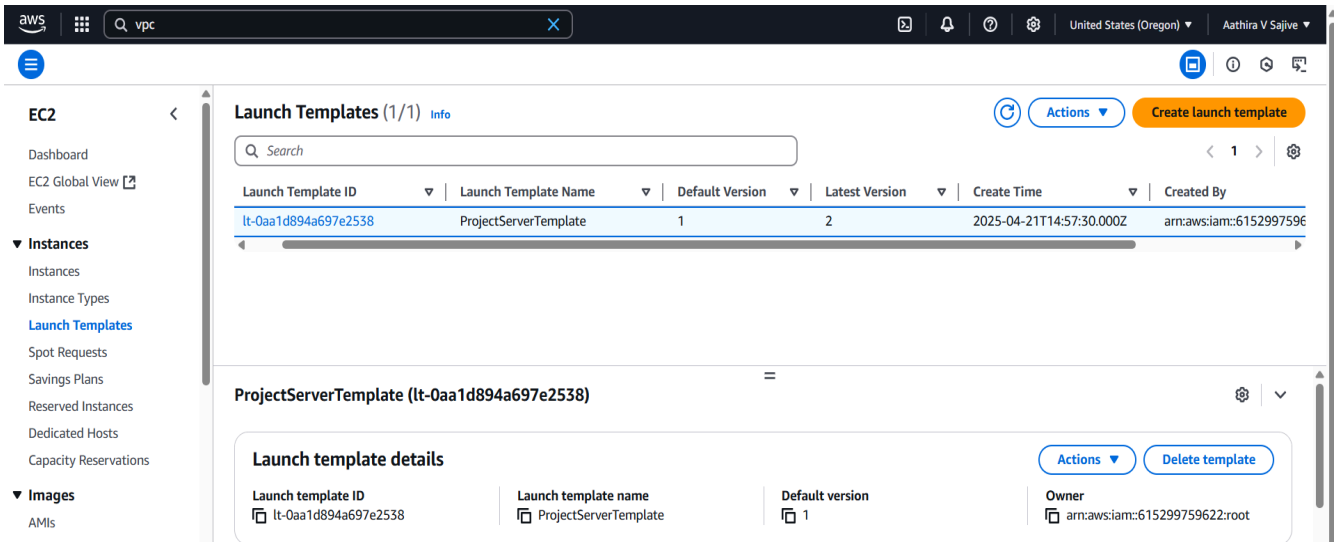
Configure Auto Scaling Group with Elastic Load Balancer

Create an AMI from Instance



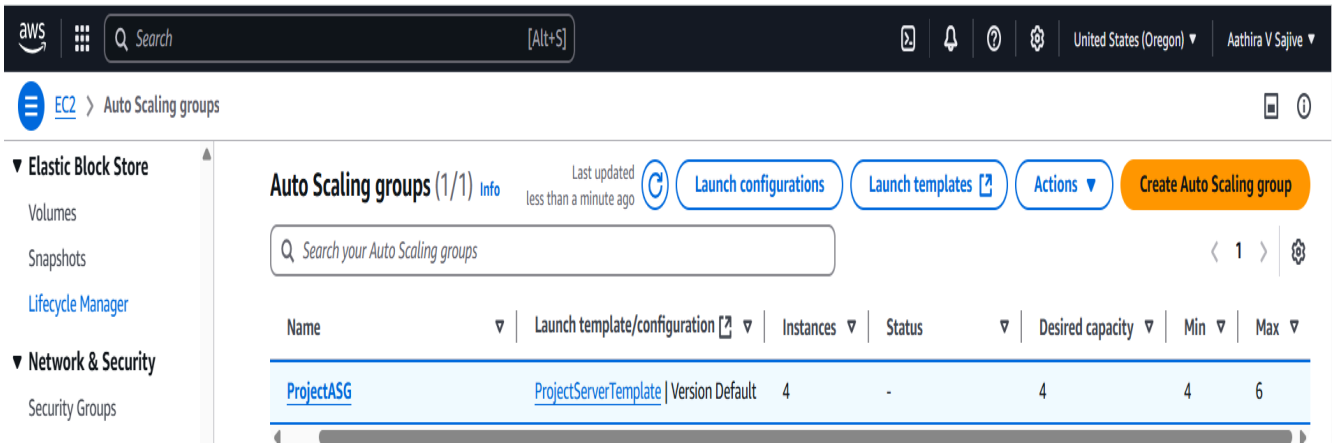
- **Go to EC2 Dashboard > Instances**
- **Select your configured instance**
- **Click Actions > Image > Create image**
- **Name: ProjectServerAMI**
- **Click Create image**

Create Launch Template



- **Go to EC2 Dashboard > Launch Templates > Create launch template**
- **Name: ProjectServerTemplate**
- **AMI: Select your AMI: ProjectServerAMI**
- **Instance type: t2.micro**
- **Key pair: Choose or create one**
- **Click Create launch template**

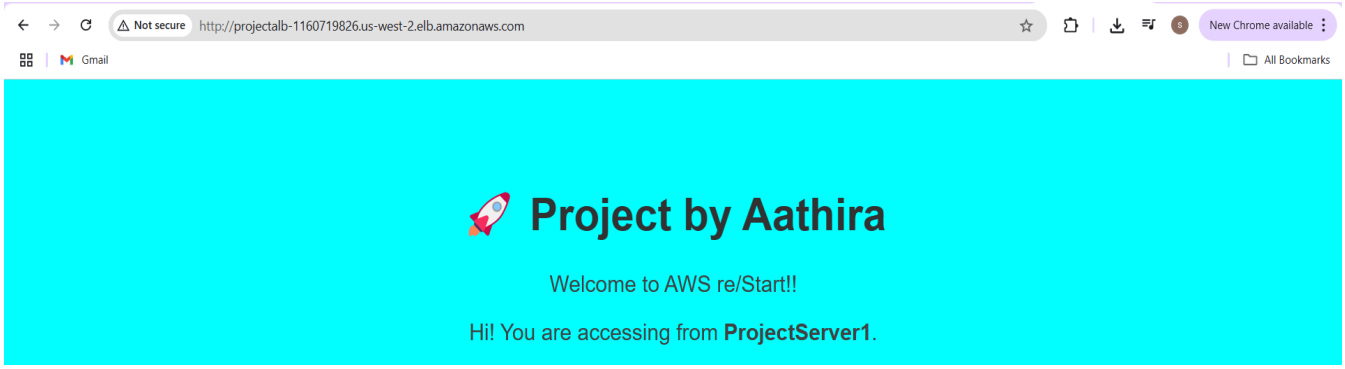
Create Auto Scaling Group



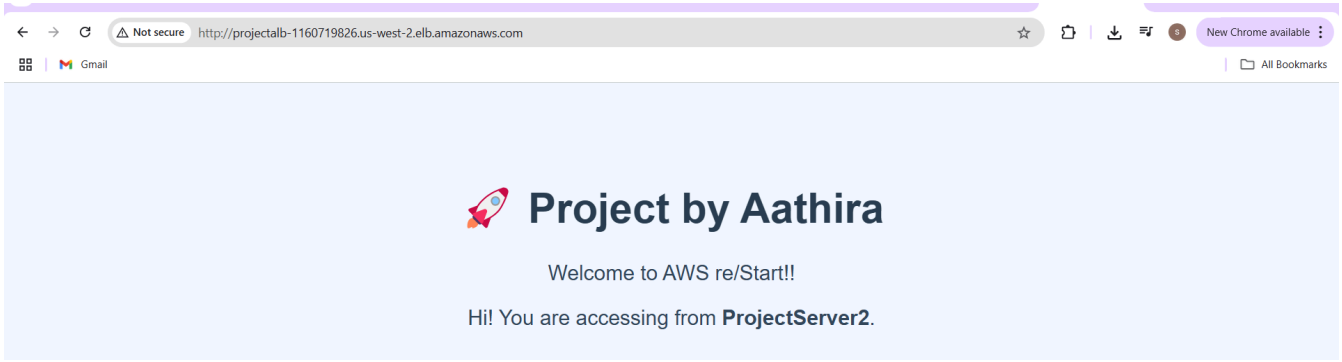
- **Go to EC2 Dashboard > Auto Scaling Groups > Create Auto Scaling group**
- **Name: ProjectASG**
- **Launch template: select ProjectServerTemplate**
- **Choose Next**
- **Choose instance launch options page, in the Network section**
- **VPC: your ProjectVPC**
- **Availability Zones and subnets:**
    - **PublicSubnet1**
    - **PublicSubnet2**
- **Choose Next**
- **Attach to an existing load balancer**
    - **Choose load balancer target groups: ProjectTargetGroup**
- **Configure group size and scaling policies**
    - **Desired capacity: `4`**
    - **Minimum capacity: `4`**
    - **Maximum capacity: `6`**
    - **Configure Health Checks**
    - **Scaling policies: Target tracking scaling policy**
    - **Metric type: Average CPU utilization.**
    - **Target value: 50**
    - **Choose Next**
    - **Add tag**
        - **Key: Name**
        - **Value: ProjectInstance**
    - **Choose Next**
    - **Click Create Auto Scaling Group**

Verify Load Balancing and Auto Scaling

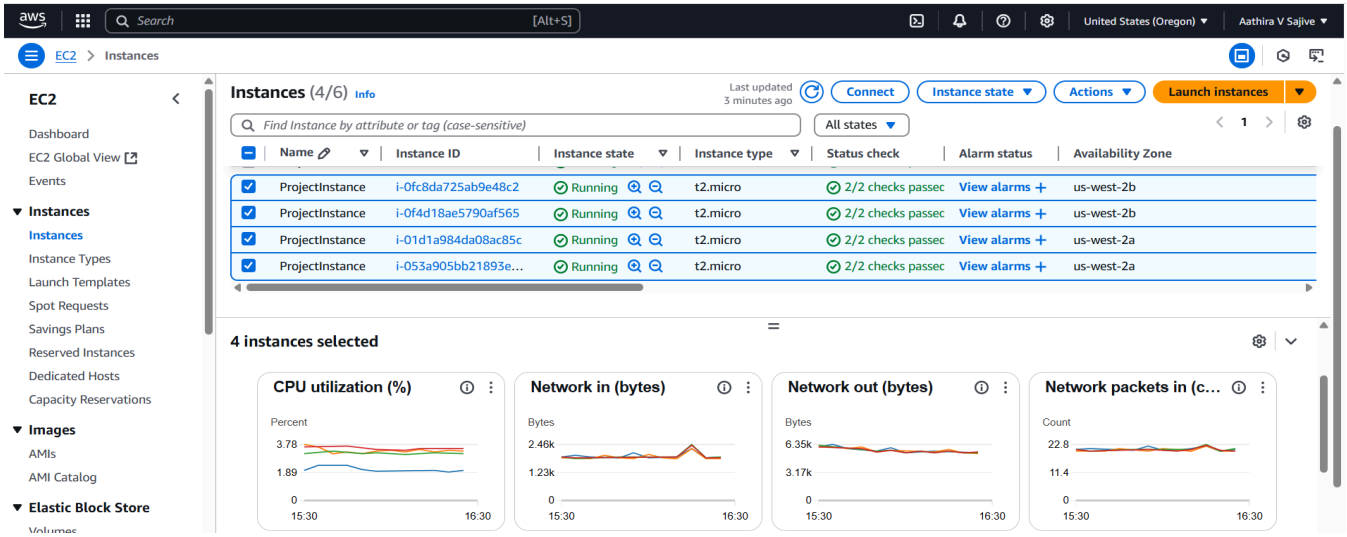
Load Balancer routed traffic to ProjectServer1



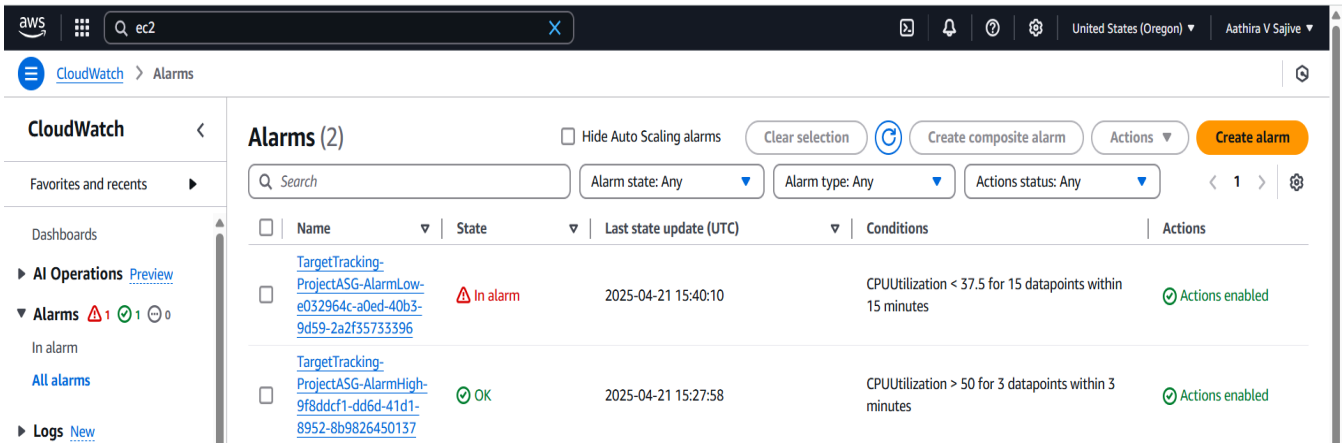
Load Balancer routed traffic to ProjectServer2



Auto Scaling Group Instances and Target Group Registration



CloudWatch Alarms for Auto Scaling Policies



CPU Utilization Graph (CloudWatch)