1. Why the Auto Scaling feature is significant in cloud computing?

1. Significance of Auto Scaling in Cloud Computing

Auto Scaling is a critical feature in cloud computing that allows applications to automatically adjust their resource allocation based on demand. This capability is significant for several reasons:

1. **Cost Efficiency**:
    - **On-Demand Resource Allocation**: Auto Scaling allows you to scale resources up or down based on real-time demand. This ensures you are only paying for the resources you actually need, avoiding over-provisioning and underutilization.
    - **Reduced Operational Costs**: By automating the scaling process, organizations can reduce the need for manual intervention, thereby lowering operational costs.
2. **Improved Performance**:
    - **Load Balancing**: Auto Scaling helps maintain optimal performance levels by distributing the workload across multiple resources. This prevents any single resource from becoming a bottleneck.
    - **Consistent User Experience**: By adjusting resources in real time, Auto Scaling ensures that applications remain responsive and performant, even during peak usage periods.
3. **Enhanced Reliability and Availability**:
    - **Fault Tolerance**: Auto Scaling can automatically replace unhealthy instances, ensuring that the application remains available and functional even in the face of failures.
    - **Redundancy**: By scaling across multiple availability zones or regions, Auto Scaling can enhance the redundancy and resilience of applications. 4. **Scalability**:
    - **Elasticity**: Auto Scaling provides the elasticity needed to handle varying workloads. It can scale out to accommodate spikes in traffic and scale in when the demand decreases.
    - **Support for Growth**: As an organization grows, Auto Scaling can seamlessly handle increased demand without requiring significant reconfiguration or manual adjustments.
5. **Automation and Management**:
    - **Reduced Manual Intervention**: By automating the scaling process, organizations can focus on core business activities rather than managing infrastructure.
    - **Simplified Management**: Cloud providers often offer intuitive dashboards and management tools to configure and monitor Auto Scaling policies, simplifying the management of cloud resources.

2. Describe the following
1. Metric-based autoscaling
2. Schedule-based autoscaling

2. Types of Auto Scaling

1. Metric-Based Autoscaling

Metric-based autoscaling, also known as dynamic autoscaling, adjusts the number of running instances based on real-time metrics and performance indicators. These metrics can include CPU utilization, memory usage, network traffic, and other custom metrics.

Key Features:

- **Dynamic Adjustment**: Automatically scales resources in response to real-time changes in demand.
- **Custom Metrics**: Users can define custom metrics tailored to their application's needs, such as request count, latency, or application-specific performance indicators.
- **Thresholds and Policies**: Users set thresholds for when to scale in or out. For example, if CPU utilization exceeds 70%, an additional instance might be launched.
- **Integration with Monitoring Tools**: Often integrates with monitoring and logging tools to gather and analyze performance data.

Advantages:

- **Real-Time Adaptation**: Provides immediate response to changing workloads, ensuring optimal performance and cost-efficiency.
- **Flexibility**: Allows for fine-tuning based on a wide range of metrics, offering granular control over scaling actions.

Disadvantages:

- **Complexity**: Setting up and managing metric-based autoscaling can be complex and may require careful tuning to avoid unnecessary scaling actions. ● **Latency**: There might be a delay between the detection of a metric threshold breach and the actual scaling action.

2. Schedule-Based Autoscaling

Schedule-based autoscaling adjusts the number of running instances based on a predefined schedule. This type of autoscaling is useful for predictable workloads that follow a regular pattern.

Key Features:

- **Predefined Schedules**: Users set specific times and dates for scaling actions. For example, an application might scale out at 9 AM on weekdays and scale in at 6 PM.
- **Predictability**: Ideal for applications with predictable usage patterns, such as business applications that see higher usage during working hours.
- **Ease of Configuration**: Typically simpler to configure than metric-based autoscaling, as it doesn't require monitoring and analyzing performance metrics.

Advantages:

- **Simplicity**: Easy to set up and manage, requiring minimal configuration compared to metric-based autoscaling.
- **Predictable Costs**: Helps in planning and predicting costs, as scaling actions occur at known times.

Disadvantages:

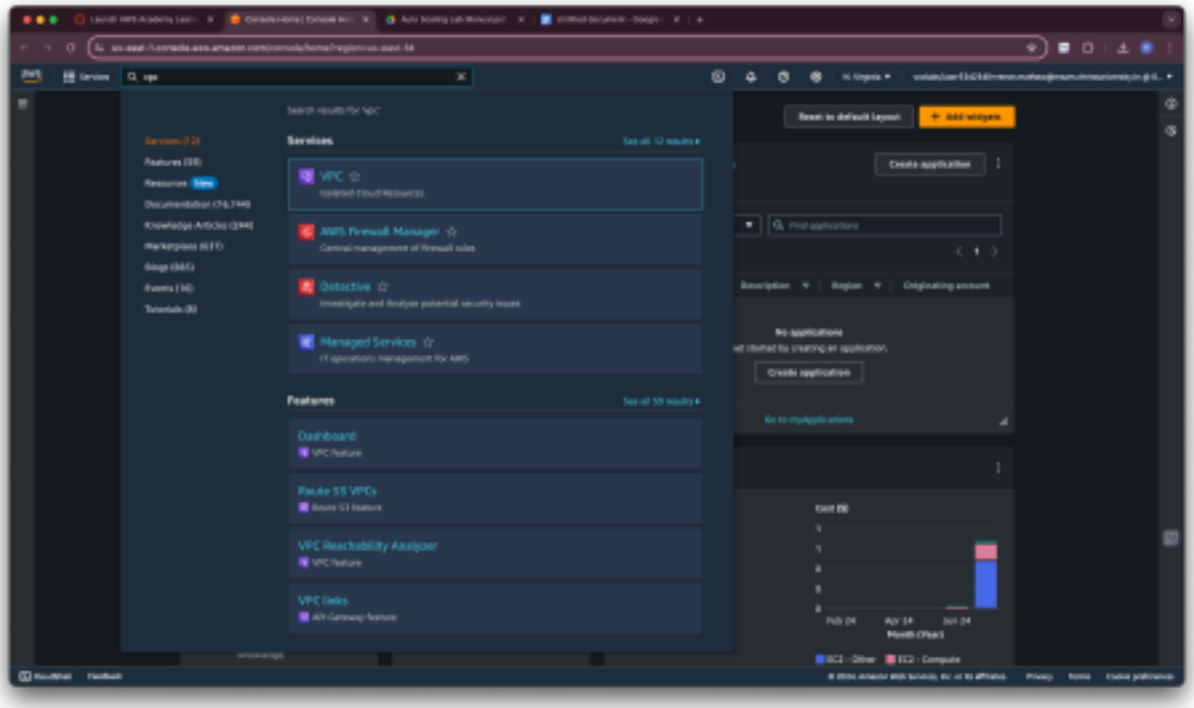
- **Inflexibility**: Not suitable for applications with unpredictable or highly variable workloads, as it doesn't respond to real-time changes in demand.
- **Potential for Over/Under Provisioning**: If usage patterns change, schedule-based autoscaling might lead to over-provisioning (and higher costs) or under-provisioning (and performance degradation).

In summary, auto scaling, whether metric-based or schedule-based, plays a crucial role in managing cloud resources efficiently. Metric-based autoscaling offers dynamic, real-time adjustments based on performance metrics, while schedule-based autoscaling provides a simple and predictable approach to resource management for applications with regular usage patterns.

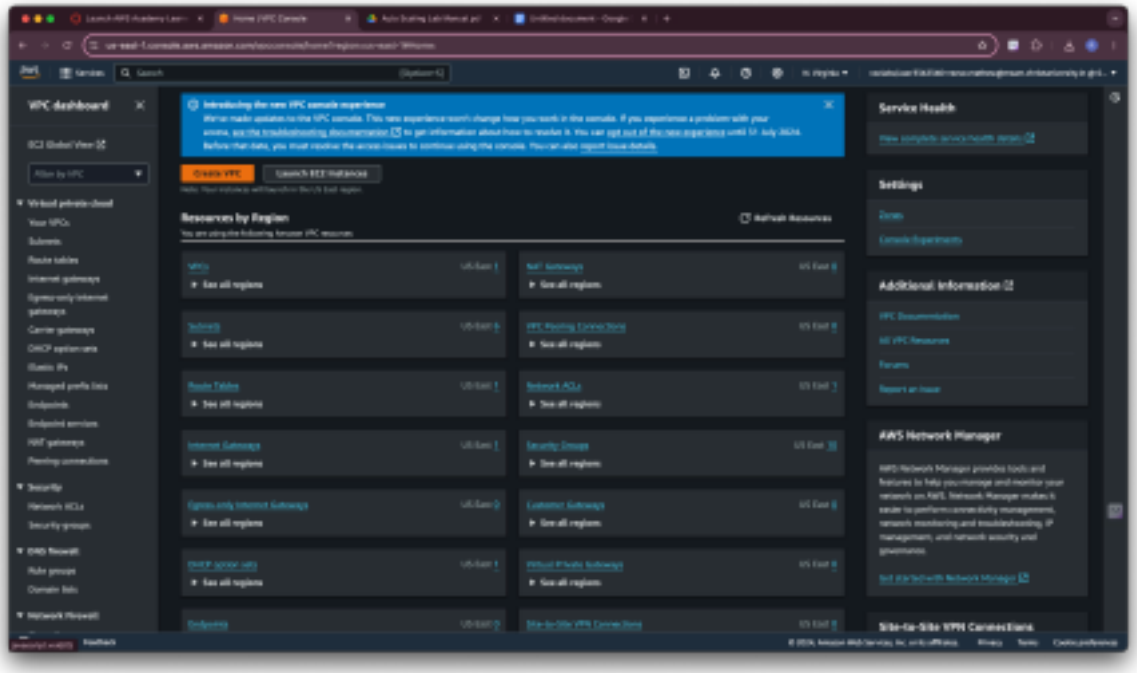
3. Demonstrate Metric based autoscaling or Schedule based auto scaling to cater your organizations business requirements. (Specify the requirements)

Auto Scaling Lab Guide Lab – 3

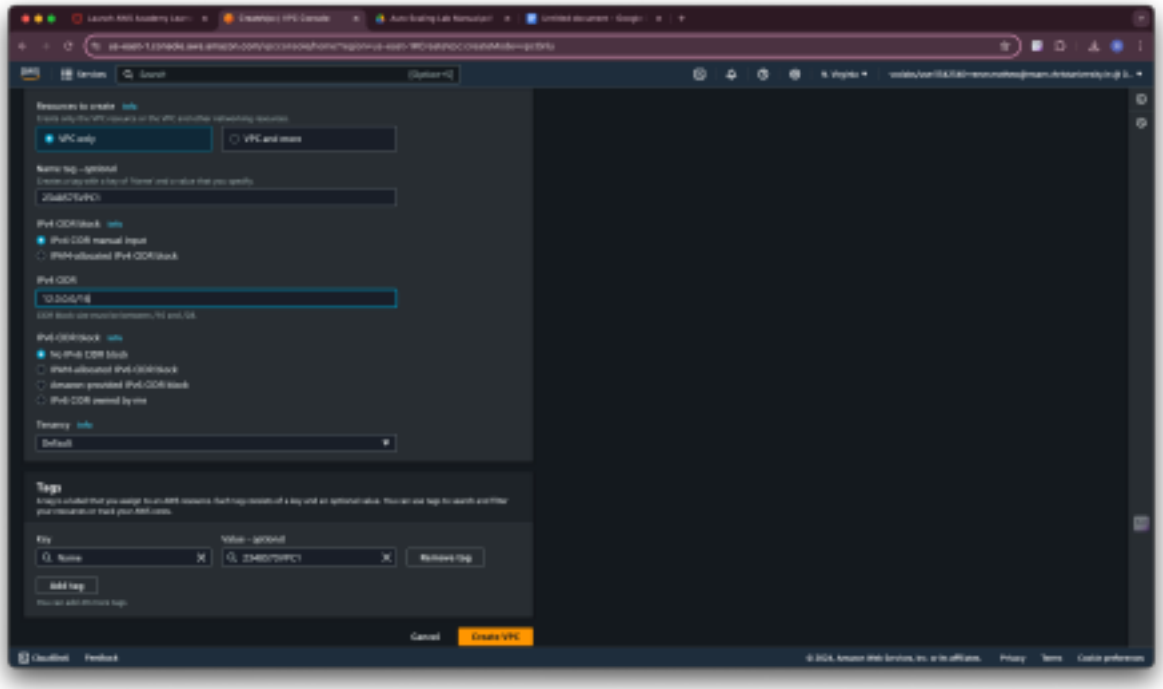
Step 1: Navigate to AWS Services
Step 2: Search for VPC (Virtual Private Cloud) and select the first option of VPC

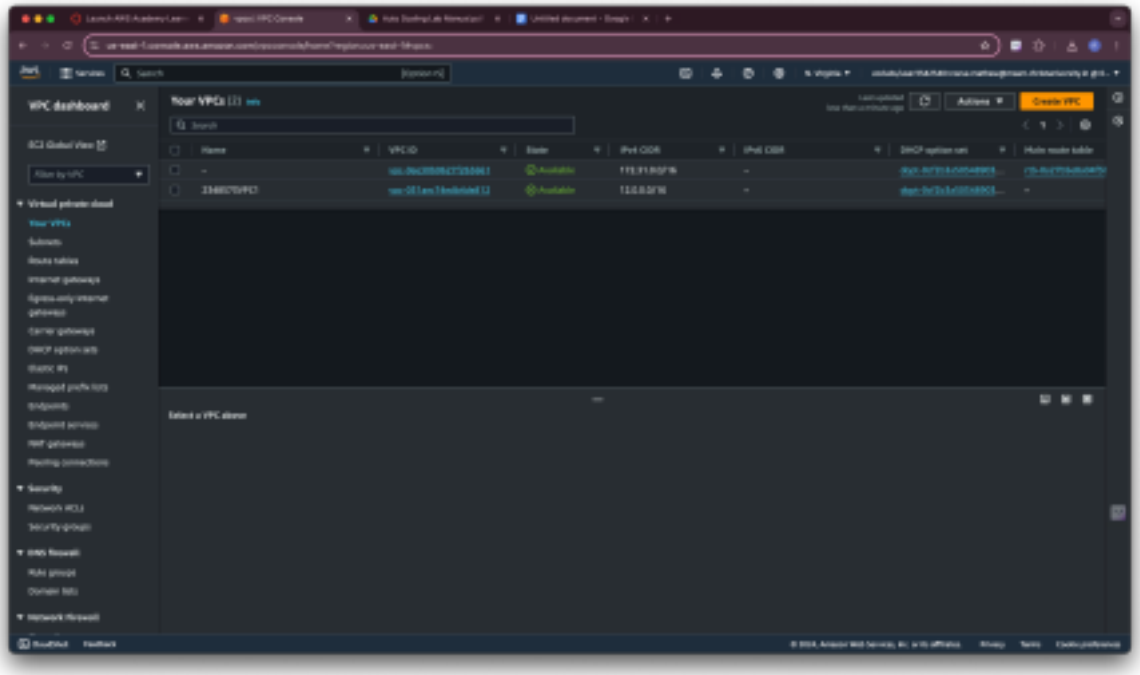


Step 3: VPC Dashboard Appears.
Click on Create VPC

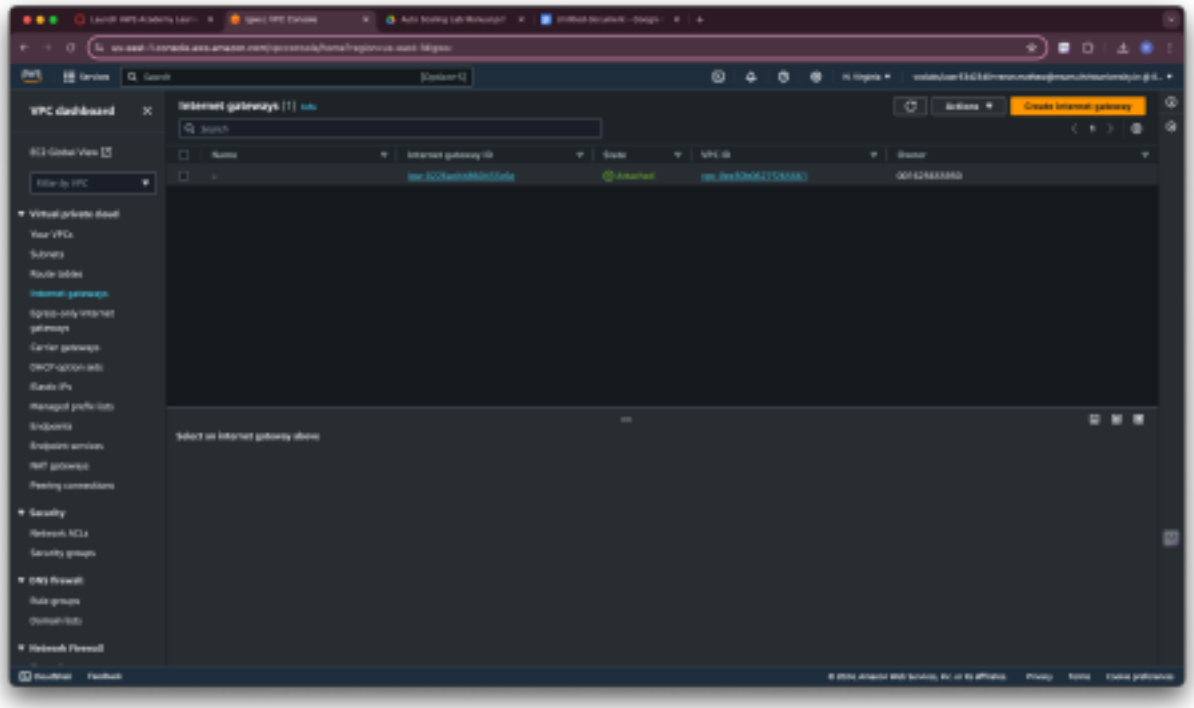


Step 4: VPC Configurations
4.1) Select VPC only.
4.2) Give a name to your VPC (in this case, we have given 2348573VPC1).
4.3) In the IPv4 text field Add 12.0.0.0/16
Leave rest all by default
4.4) Click on Create VPC

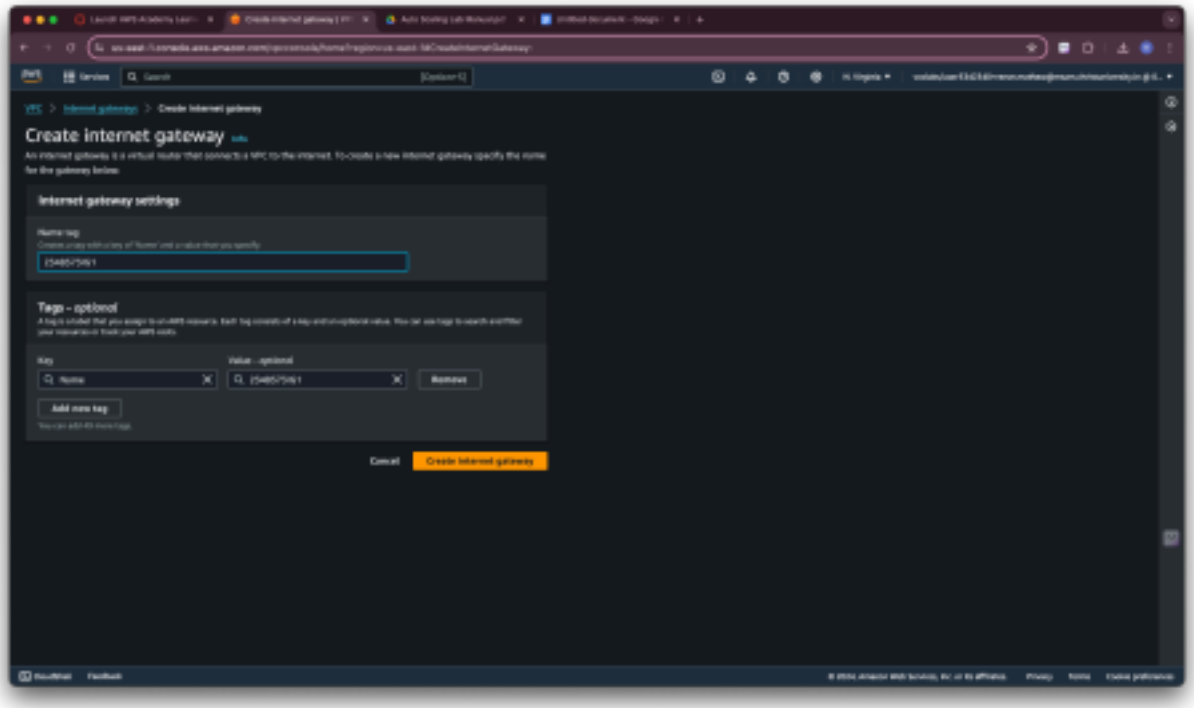




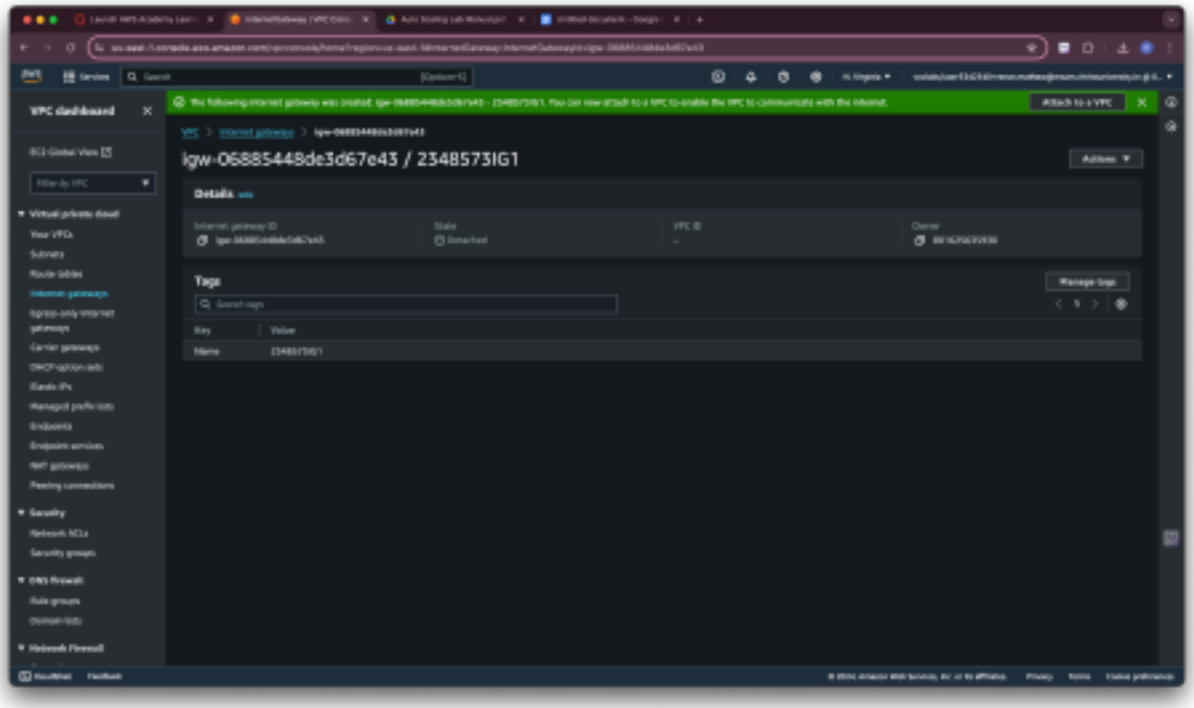
Step 5: Click on Internet Gateway at the left hand-side column
Step 6: Click on Create Internet Gateway



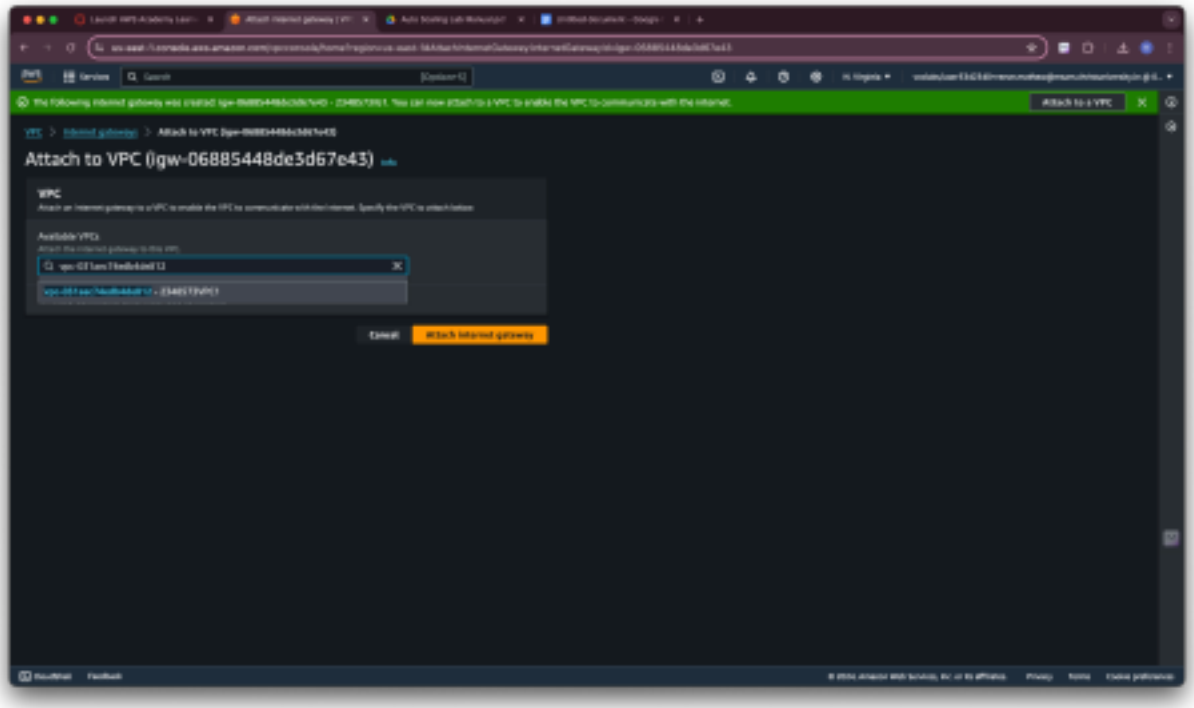
Step 7: Give your Internet Gateway a name. In this case I have given 2348573IG1 and click on Create

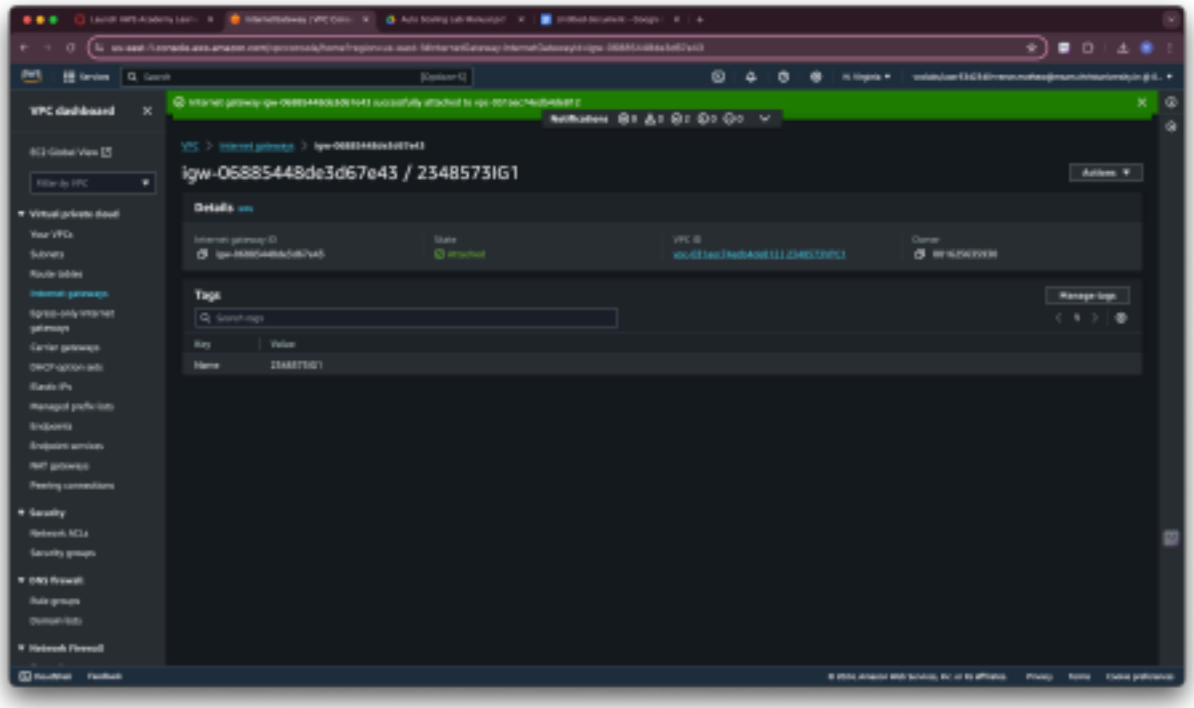


Step 8: Click on Attach to a VPC at the top right corner

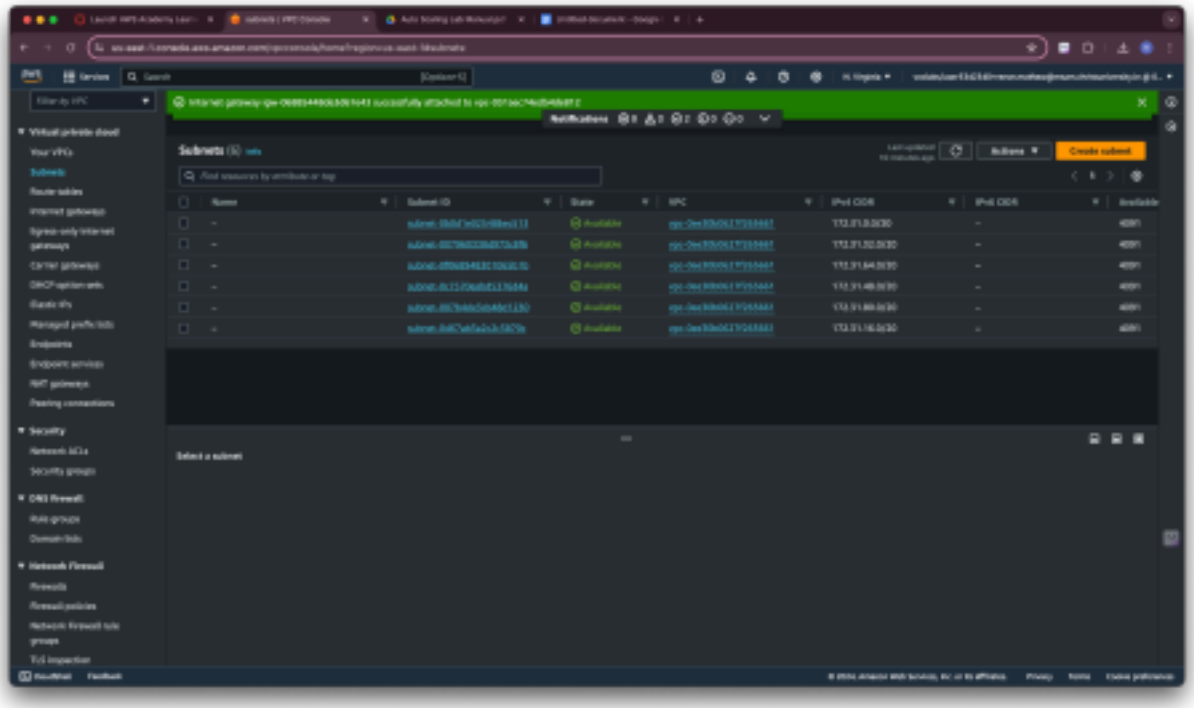


Step9: SelecttheVPCthatyoucreated. Inthiscase it is2348573VPC1andthenclickon Attachto InternetGateway

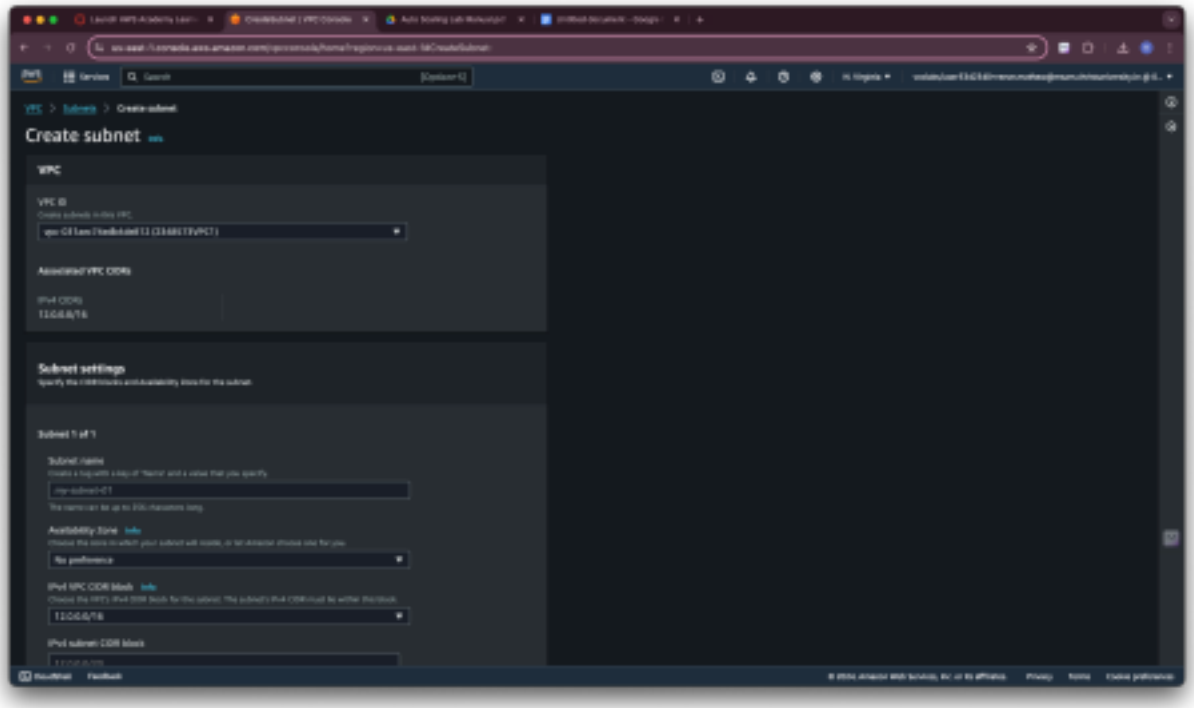




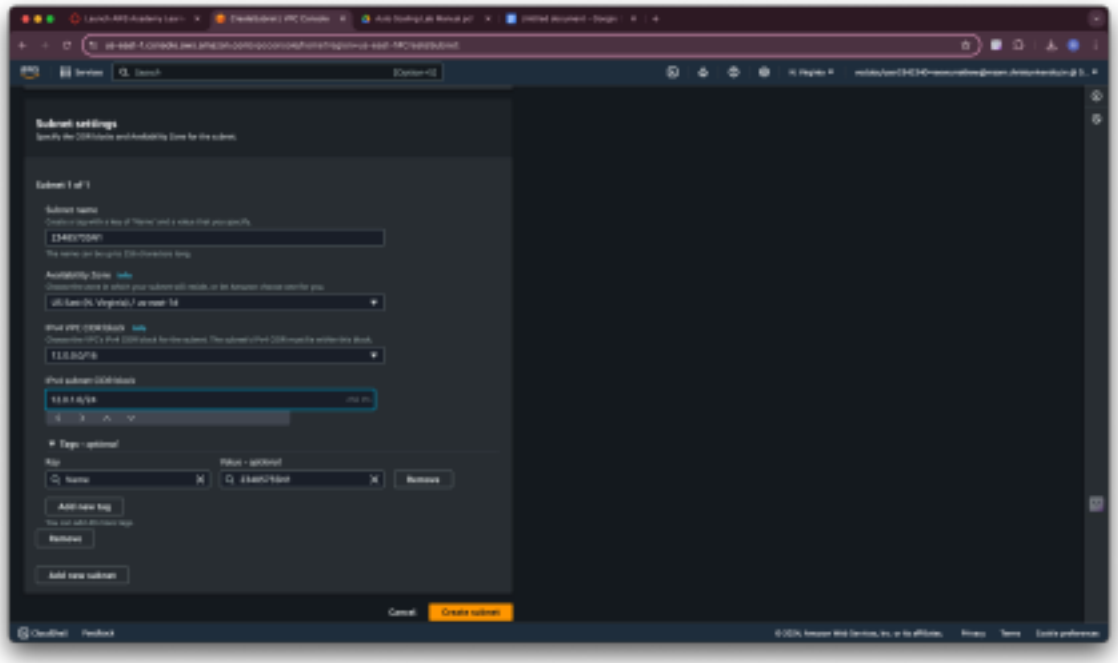
Step10: ClickonSubnetatthe lefthandsideofthedashboard
Step11: ClickonCreateSubnet



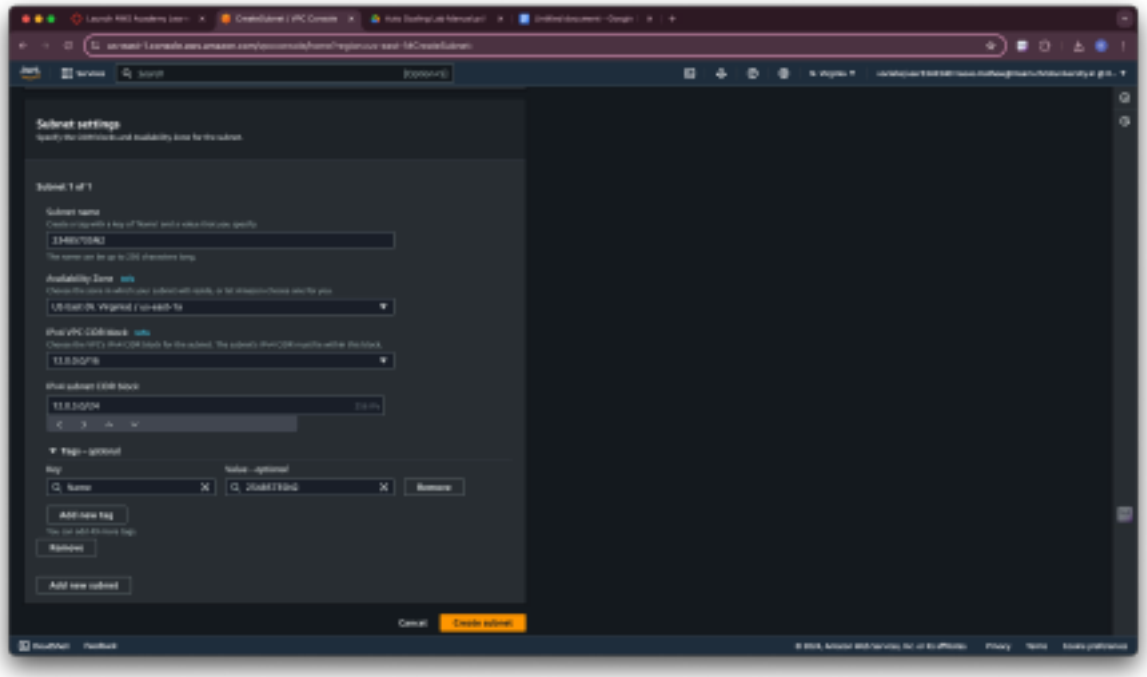
Step 12: Select the VPC that you created from the dropdown menu.



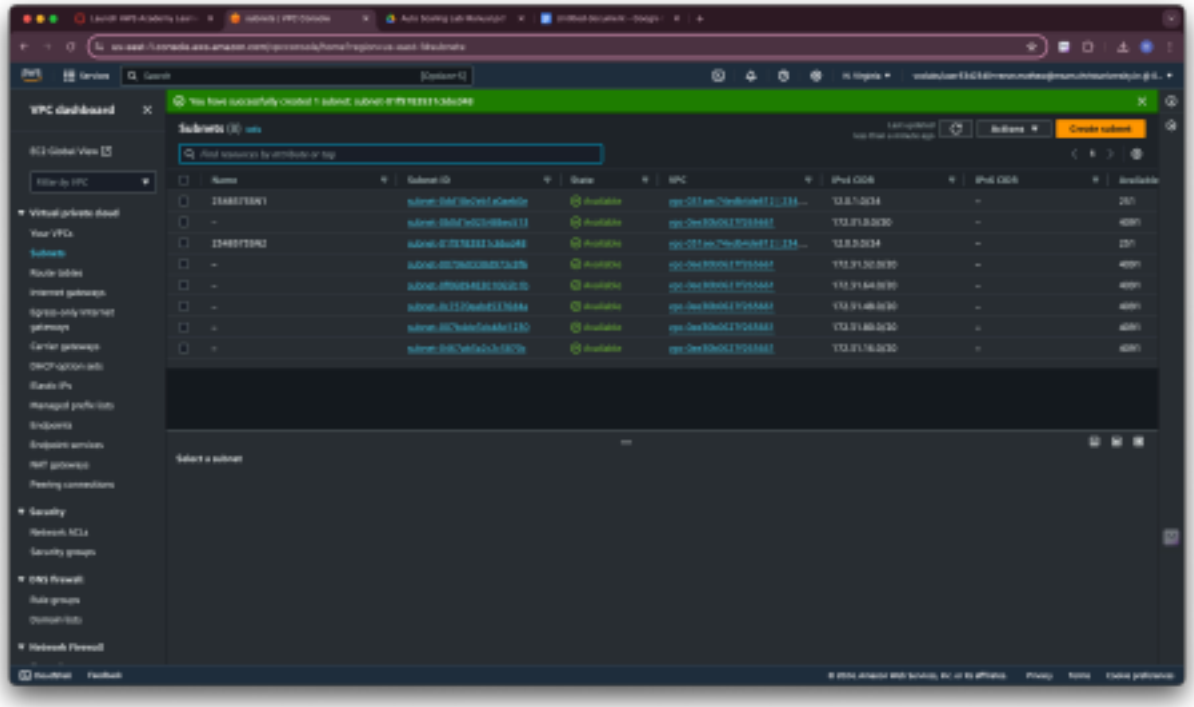
Step 13: Configuration of Subnet
13.1) Provide a name to your subnet. In this case I have given 2348573SN1
13.2) Availability Zone: us-east-1d
13.3) IPv4 subnet CIDR block: 12.0.1.0/24



Step 14: Click on Add new subnet
Step 15: Configuration for Subnet 2
15.1) Provide a name to your subnet. In this case I have given 2348573SN2
15.2) Availability Zone: us-east-1a
15.3) IPv4 subnet CIDR block: 12.0.3.0/24

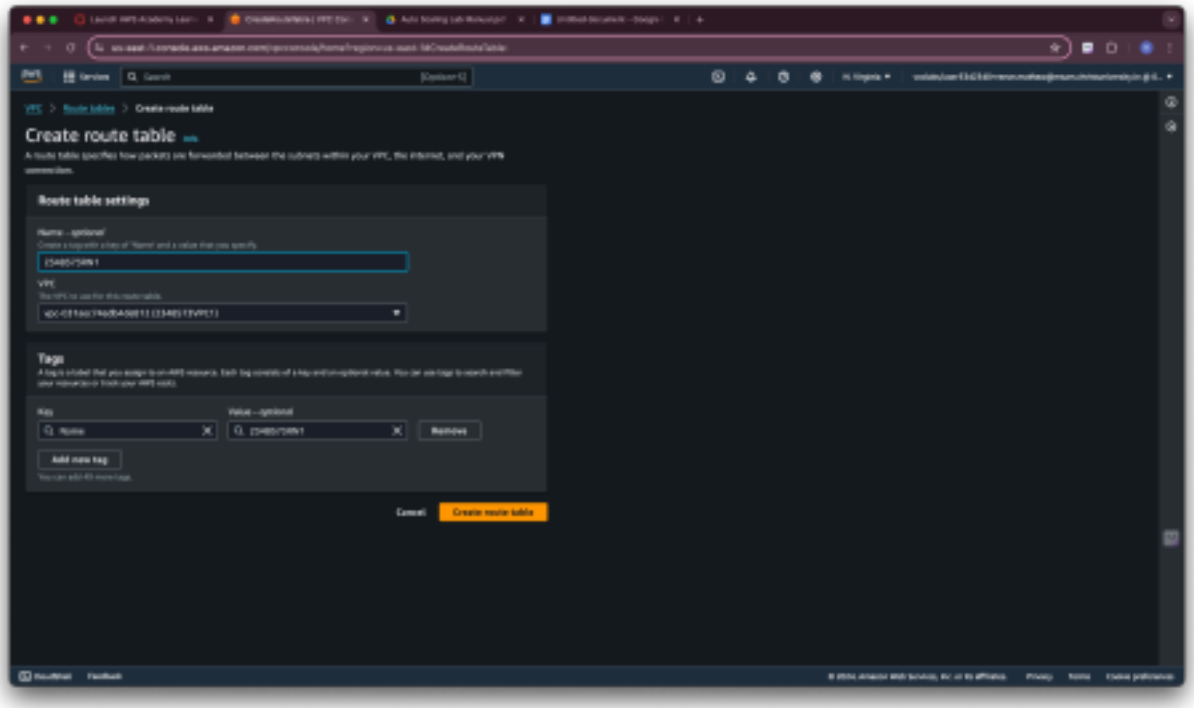


Step 16: Click on Create Subnet

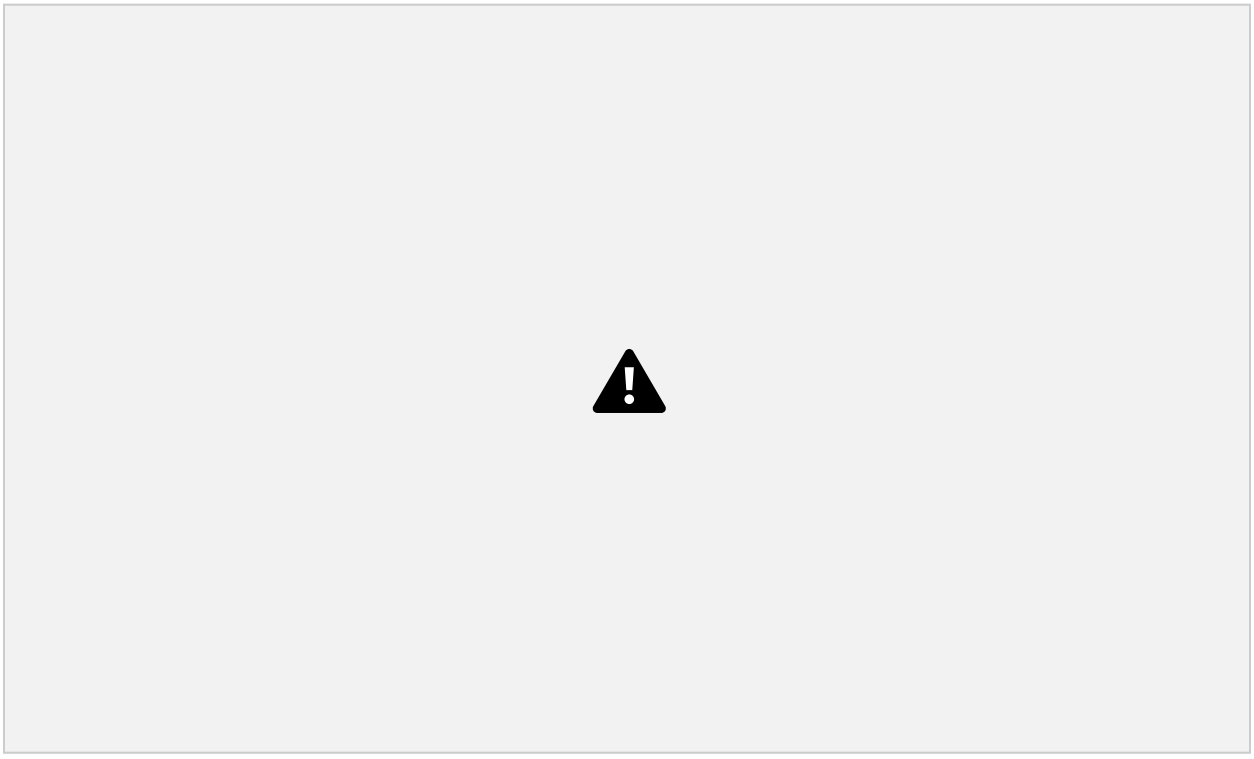


Step 17: Click on Route Table under the VPC section on the left hand side and click on Create Route Table

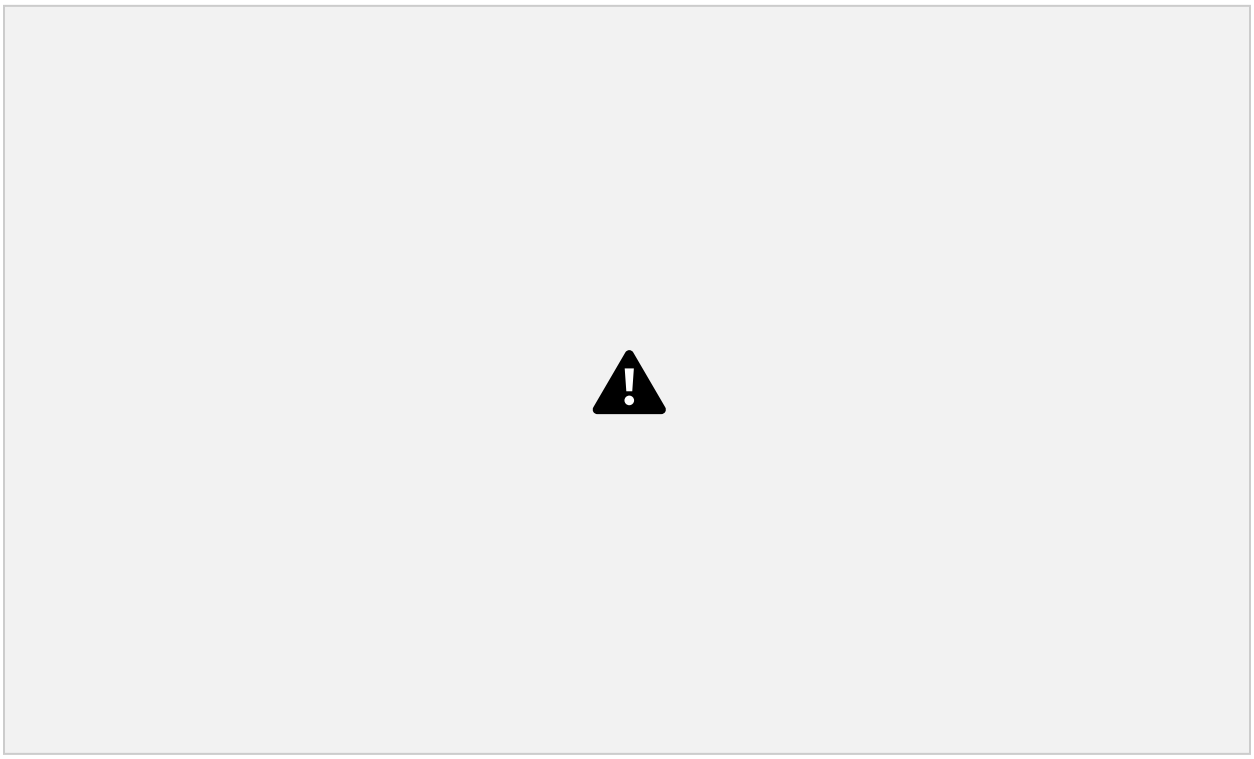
Step 18: Provide a name to the Subnet and select the VPC you have created and click on Create Route Table

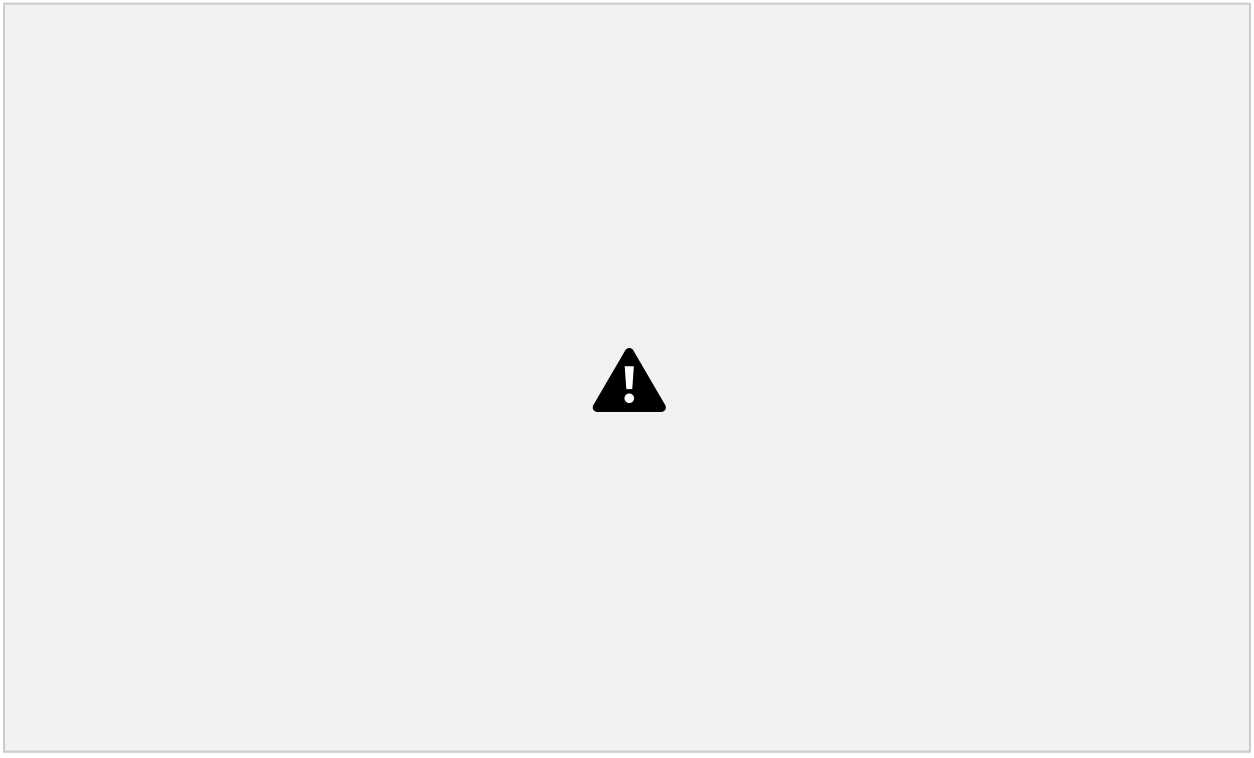


Step 19: Click on Subnet Associations and then Edit Subnet Associations Step 20: Select the two created Subnets and click on Save Associations.

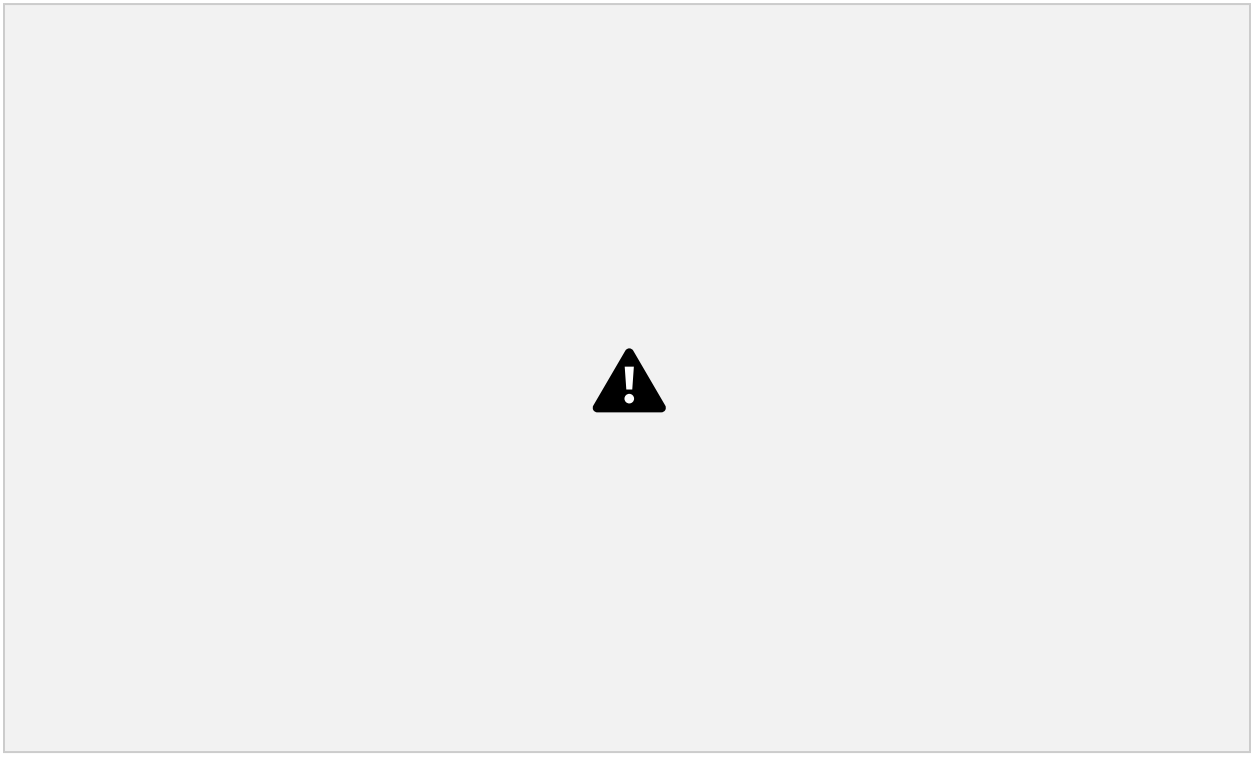


Step 21: Click on Edit Routes
Step 22: Click on Add Routes
Step 23: Routes Configurations:
23.1) Destination: 0.0.0.0/0
23.2) Select Internet Gateway for the Target
23.3) Select the created Internet Gateway from the pop-up menu
23.4) Click on Save Changes

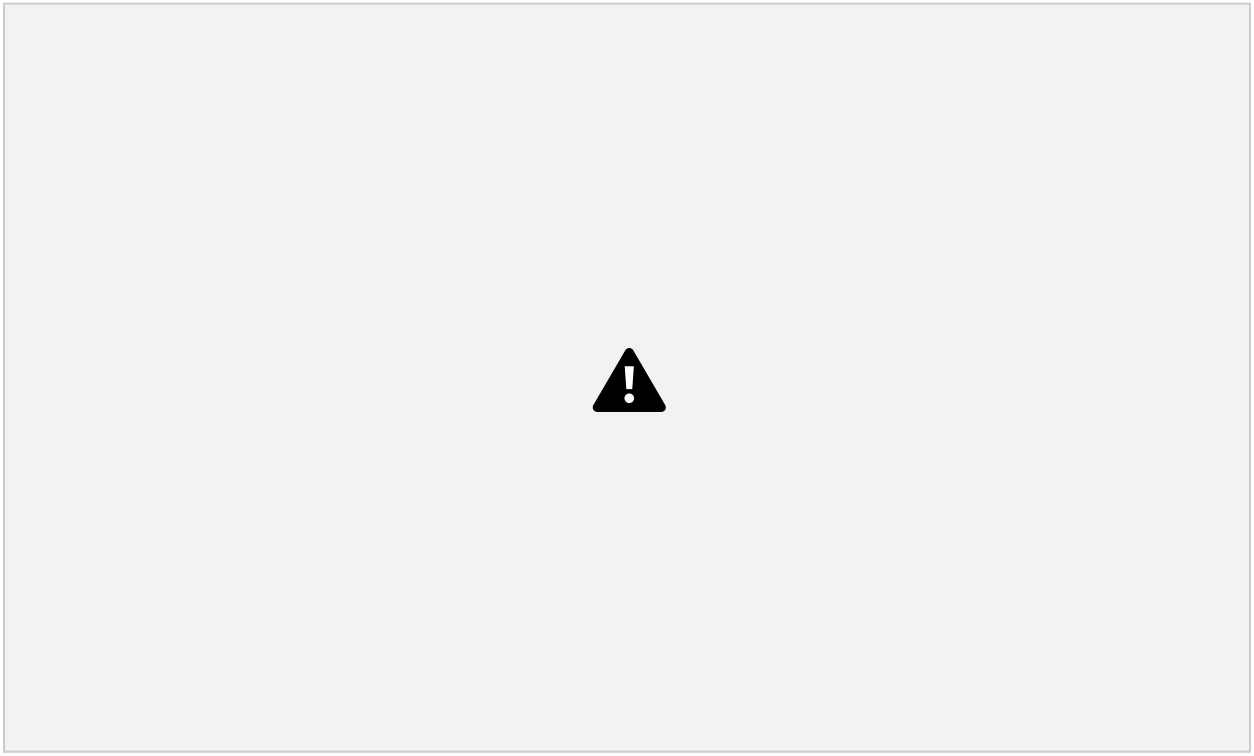




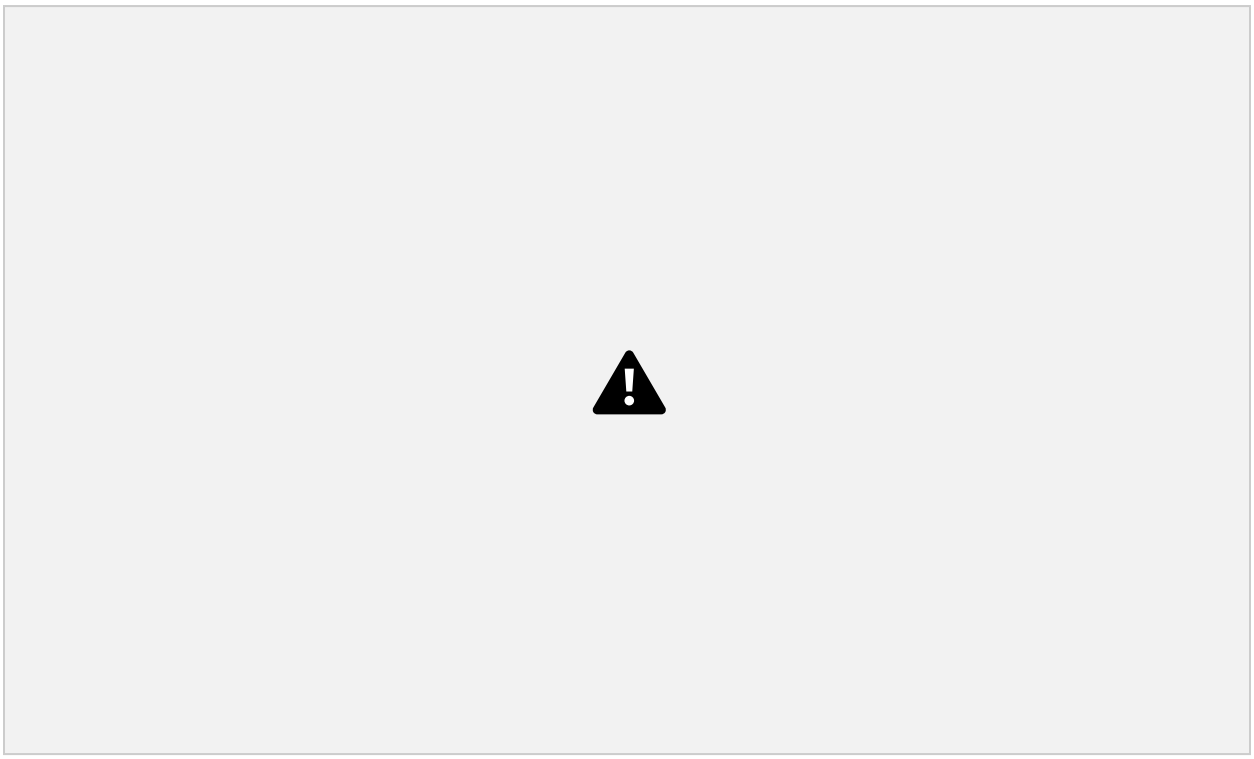
Step 24: Navigate to EC2 Services
Step 25: Under the left-hand side under Load Balancing select Target Group
Step 26: Click on Create Target Group

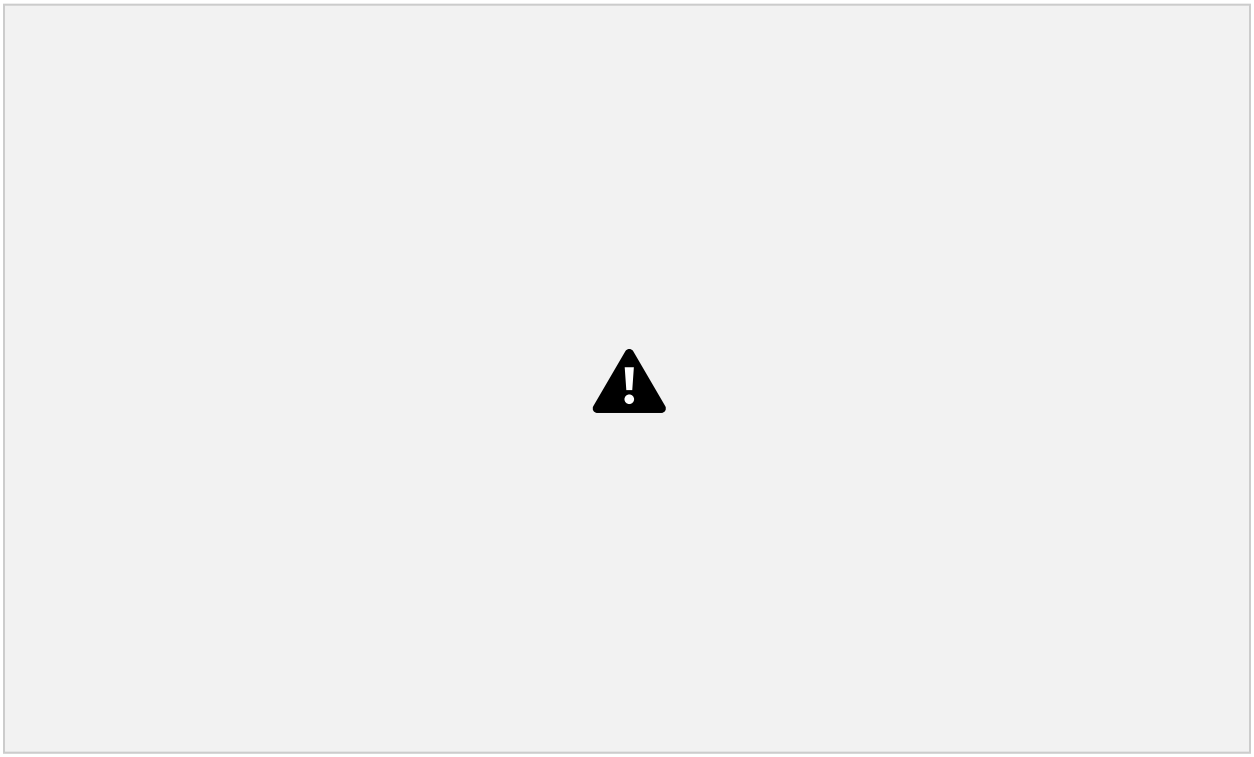


Step 27: Target Group Configurations:
27.1) Target Type: Instances (Default)
27.2) Provide a name: 2348573TG1
27.3) Select the VPC you Created, in my case (2348554VPC1)
27.2) Click on Next

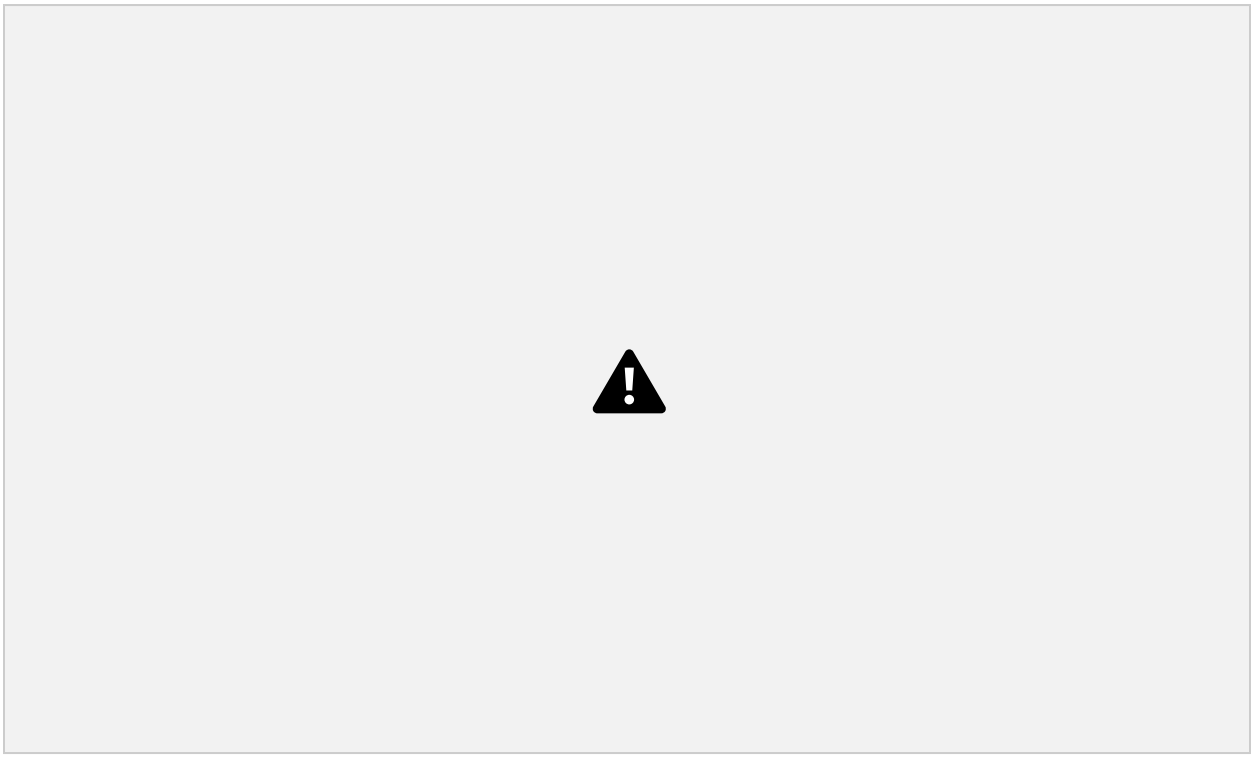


Step 28: Click on Create Target Group





Step 29: Navigate to Load Balancers at the left-hand side tab
Step 30: Click on create Load Balancer



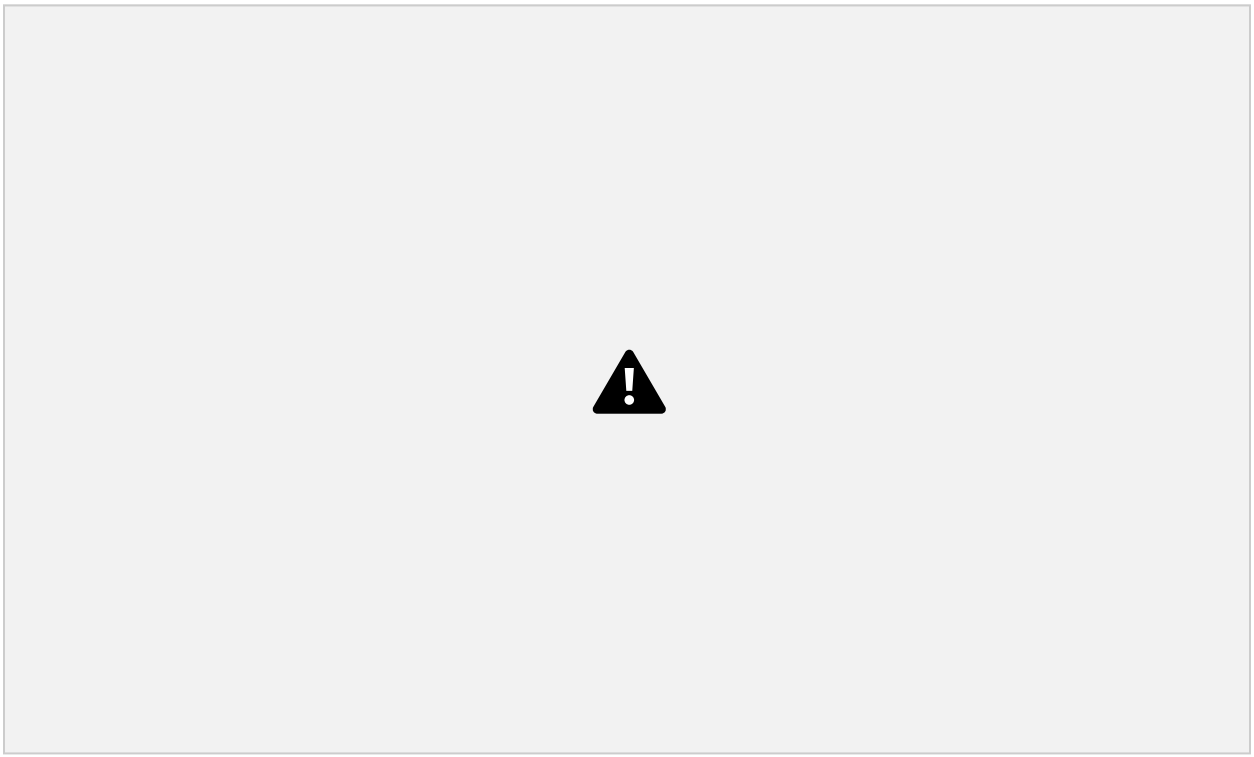
Step 31: Select Application Load Balancer
Step 32: Load Balancer Configurations
Name: 2348573LB1
Scheme: Internet-facing
Load Balancer IP address type: IPv4

Step 33: Select the VPC you created and select both the subnets
Step 34: Click on Create a new security Group

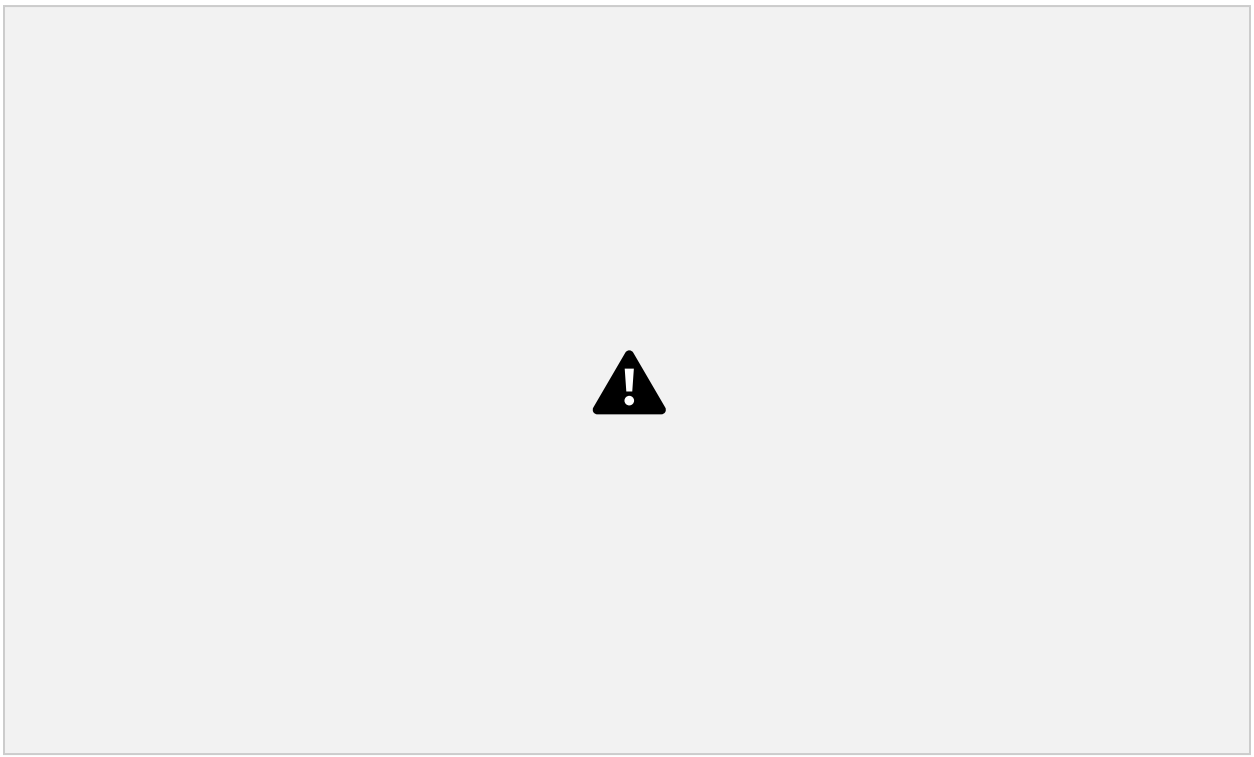
Step 35: Security Group Configurations
Name: 2348573SG1
Description: Allowing required rules and ports
VPC: Select the VPC you have created from the drop down menu

Step 36: Click on Add Inbound Rules
36.1) Add SSH and IP as 0.0.0.0/0
36.1) Add HTTP and IP as 0.0.0.0/0
36.1) Add HTTPS and IP as 0.0.0.0/0

Step 37: Click on Create Security Group

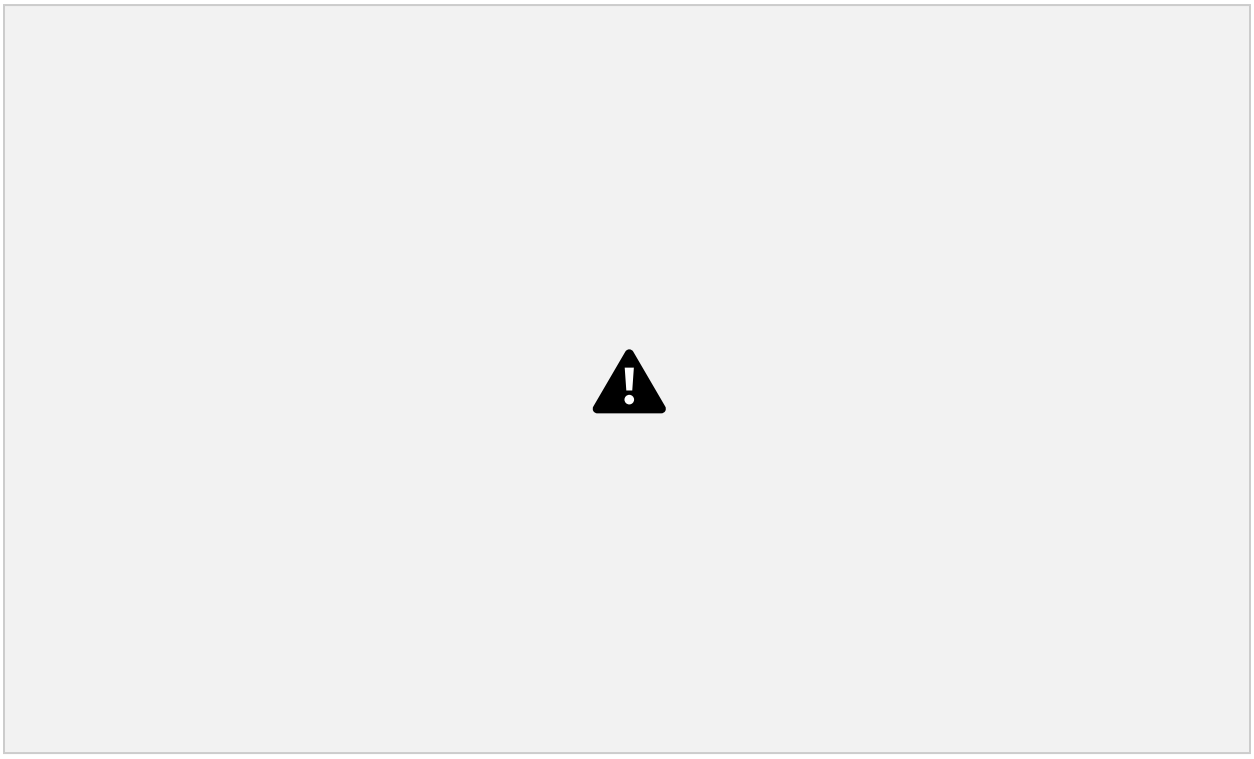


Step 38: Close the window and go back to the Load Balancer page

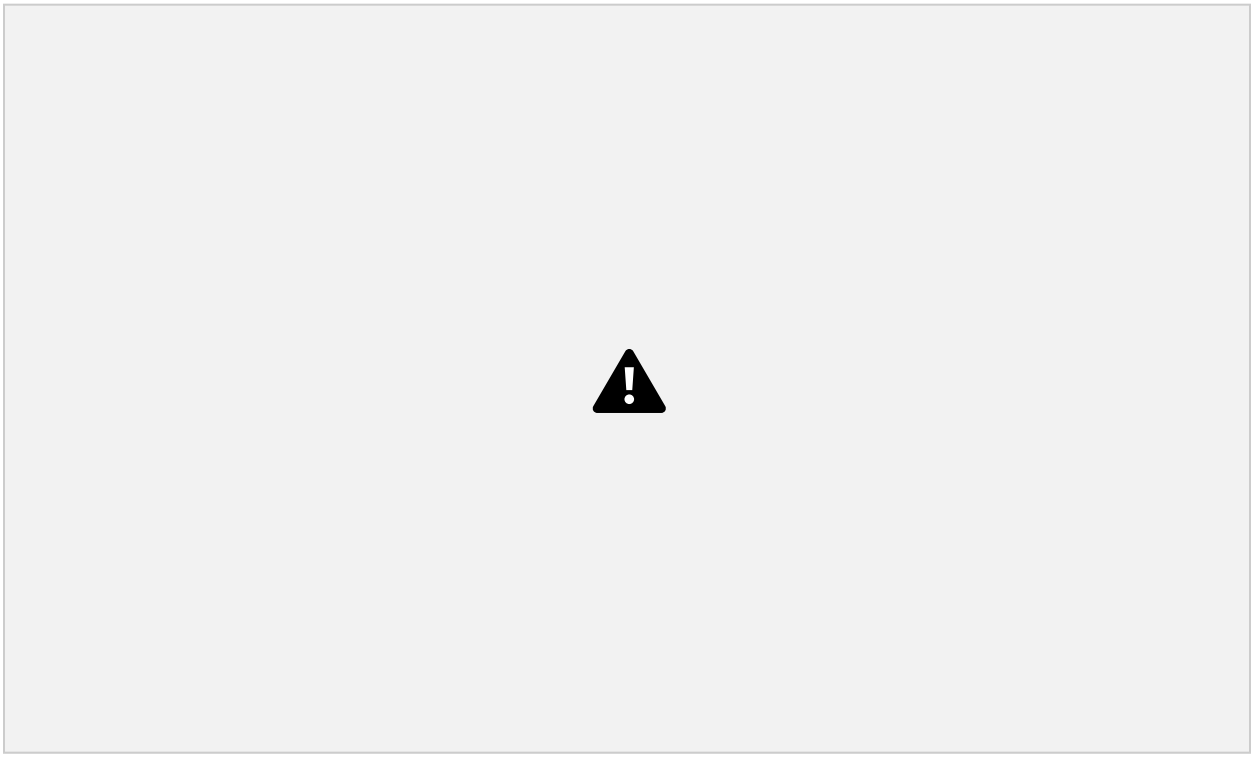


Step 39: Create the Selected Target Group

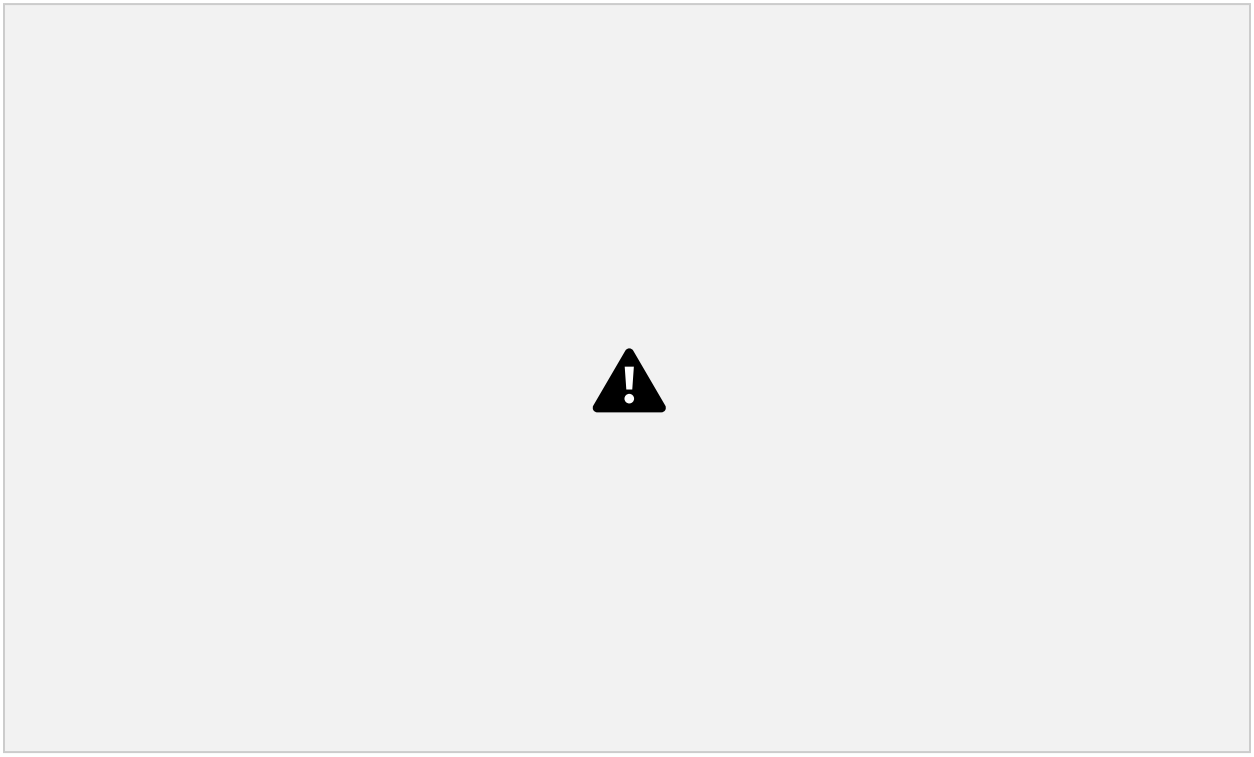
Step 40: Click on Create Load Balancer



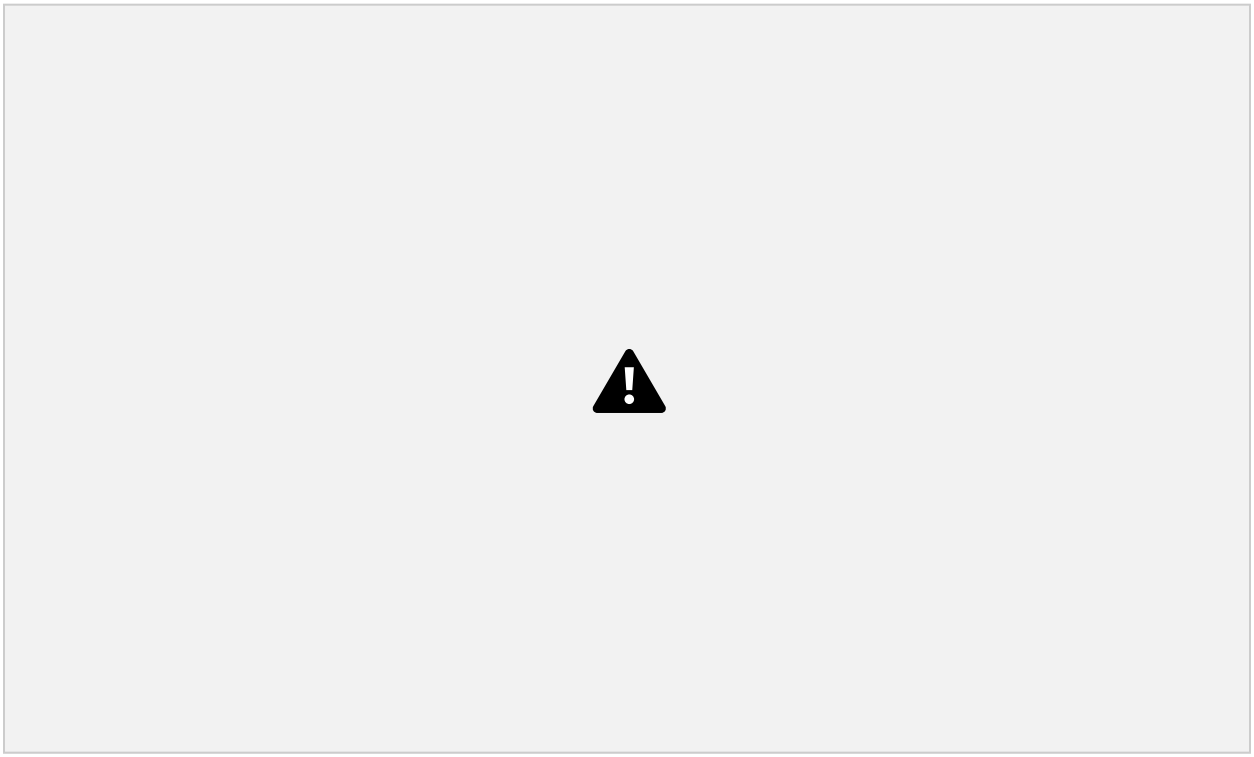
Step 41: Wait for Load Balancer state to turn to active from provisioning

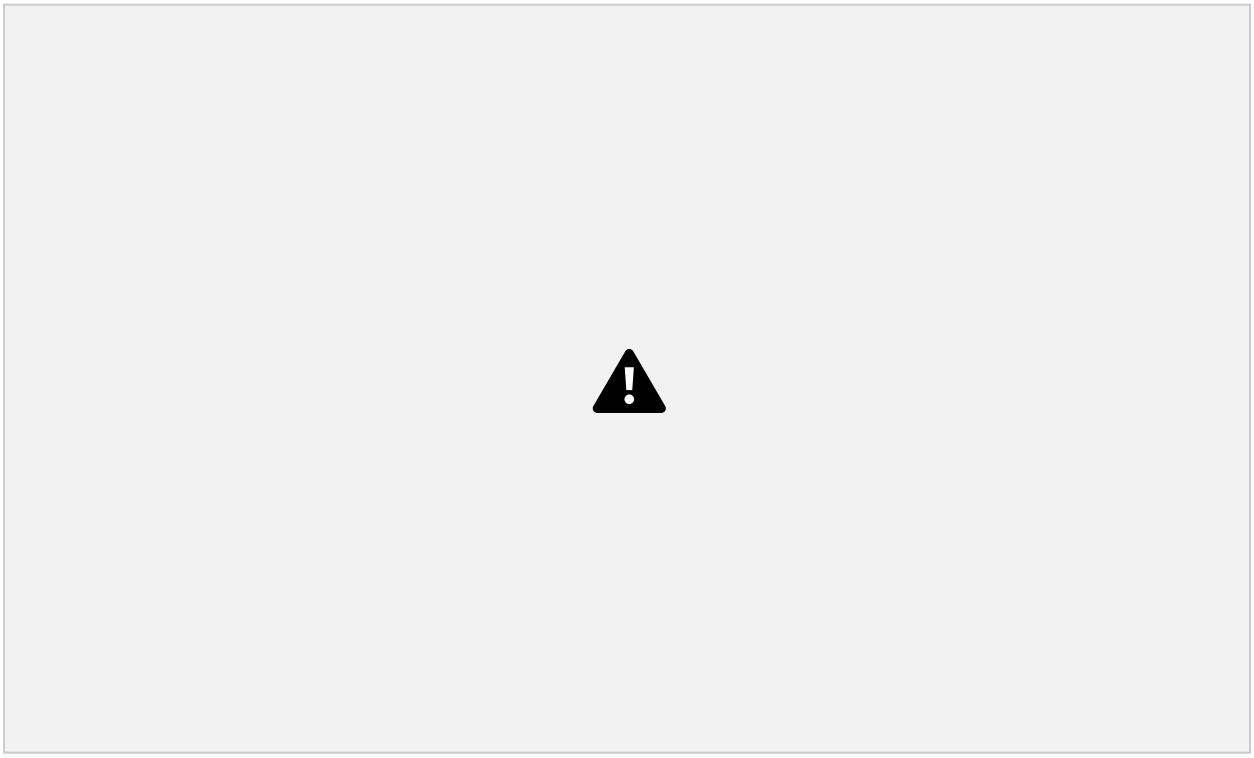


Step 42: From the left-hand bar navigate to Auto Scaling Groups



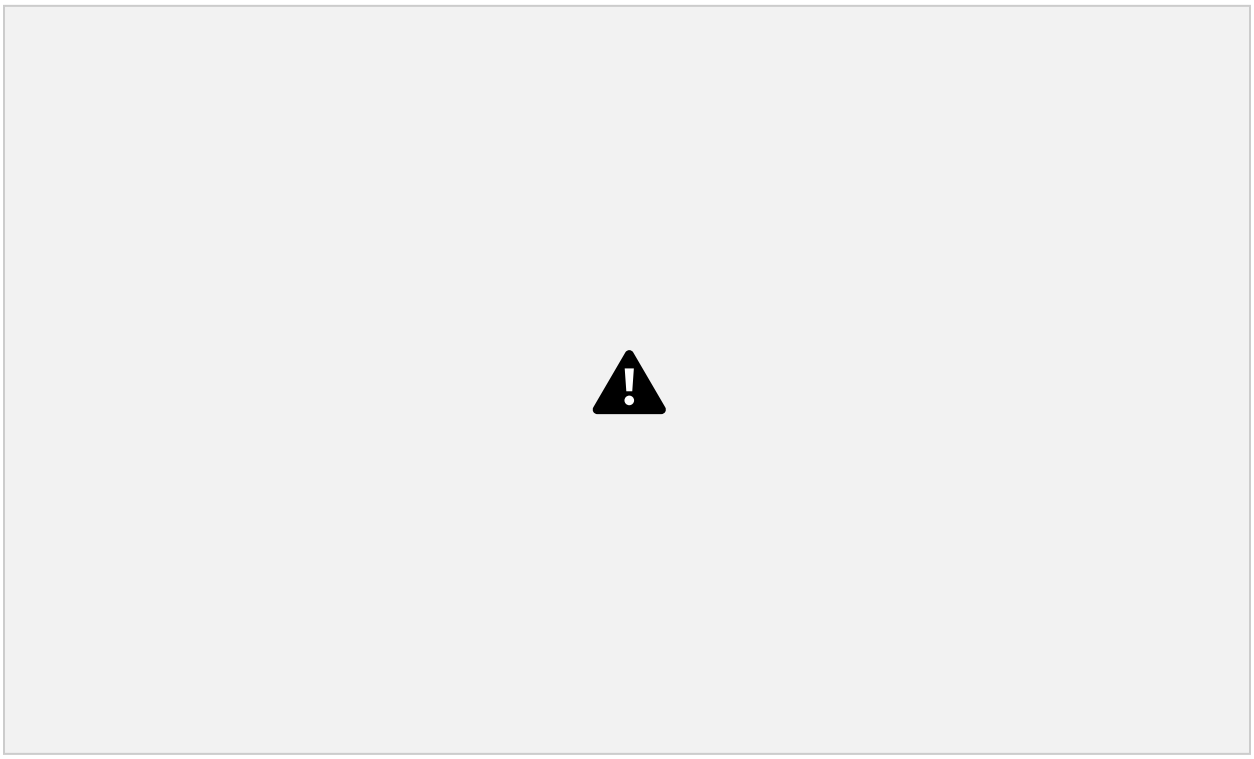
Step 44: Provide a name to the Auto Scaling Group: 2348573ASG1
Step 45: Click on Create a Launch Template



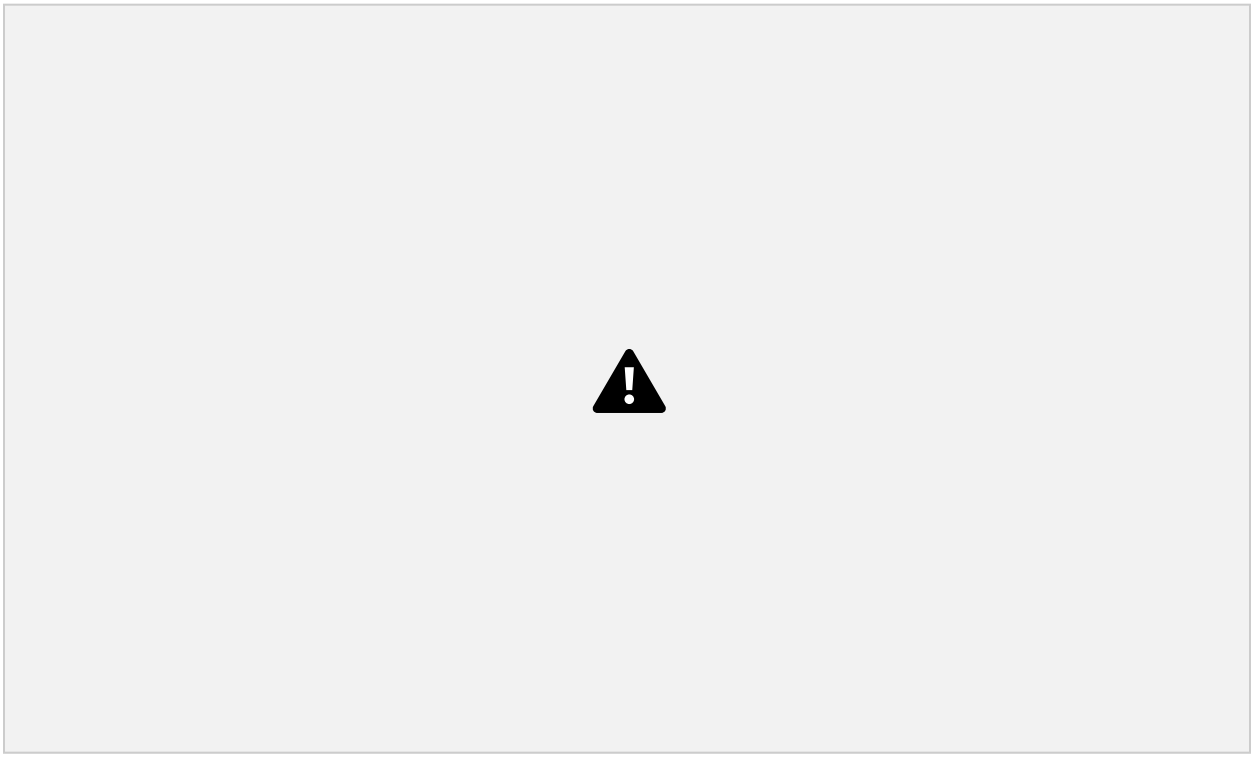


Step 46: Configuring the Launch Template
46.1) AMI: Amazon Linux 2023 AMI 2023.5.20240708.0 x86_64 HVM kernel-6.1
46.2) Instance Type: t2.micro
46.3) Key Pair: Vockey
46.4) Within Subnet select “Don’t include in Launch template”
46.4) Security Group: 2348554SG1
46.6) Within Advanced Network Configuration Enable Auto Assign Public IP 46.5) Navigate to Advanced details and within the User data type the below content

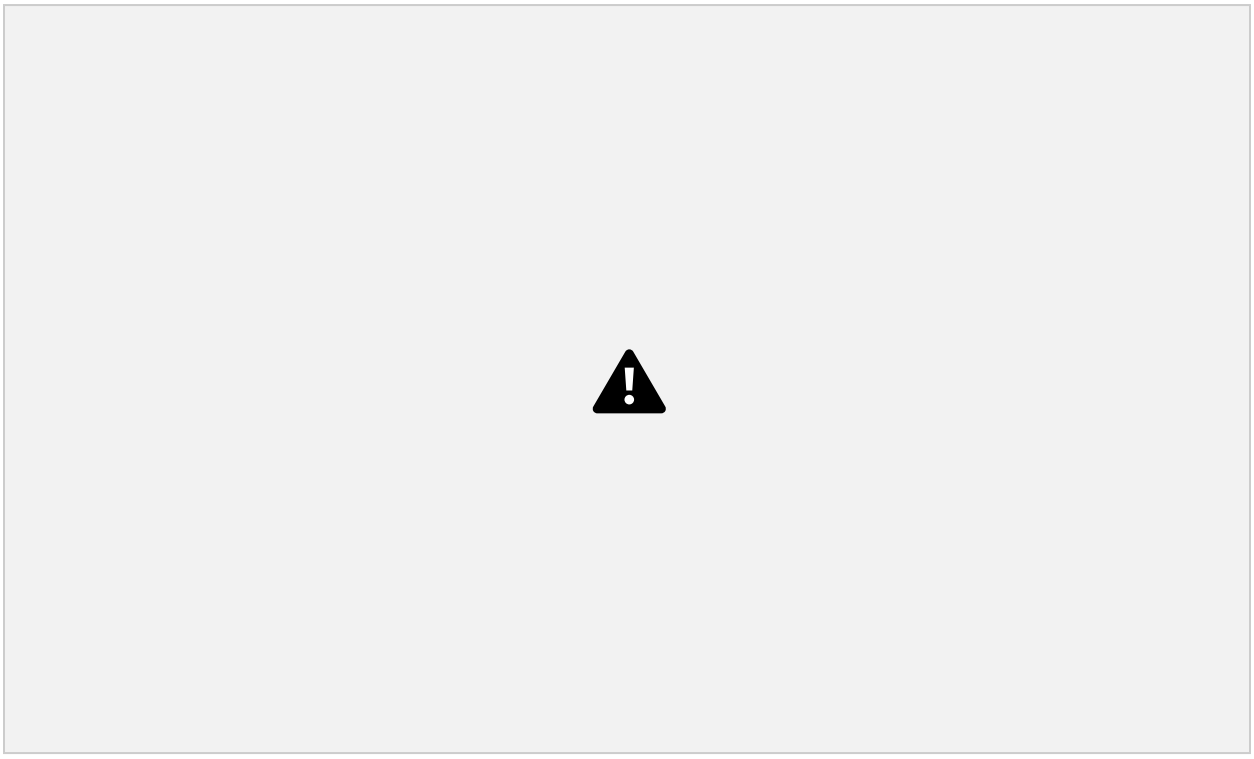
```
#!/bin/bash
# Update the package repository
yum update -y
# Install httpd (Apache)
yum install -y httpd
# Start the httpd service
systemctl start httpd
# Enable the httpd service to start on boot
systemctl enable httpd
# Get the hostname
HOSTNAME=$(hostname)
# Create an index.html file that prints the hostname in the heading
echo "<html><body><h1>Hostname: $HOSTNAME</h1></body></html>" >
/var/www/html/index.html
# Restart the httpd service to apply changes
service httpd restart
```



Step 47: Click on Create Launch template
Step 48: Navigate back to Auto Scaling Group and choose the created Template

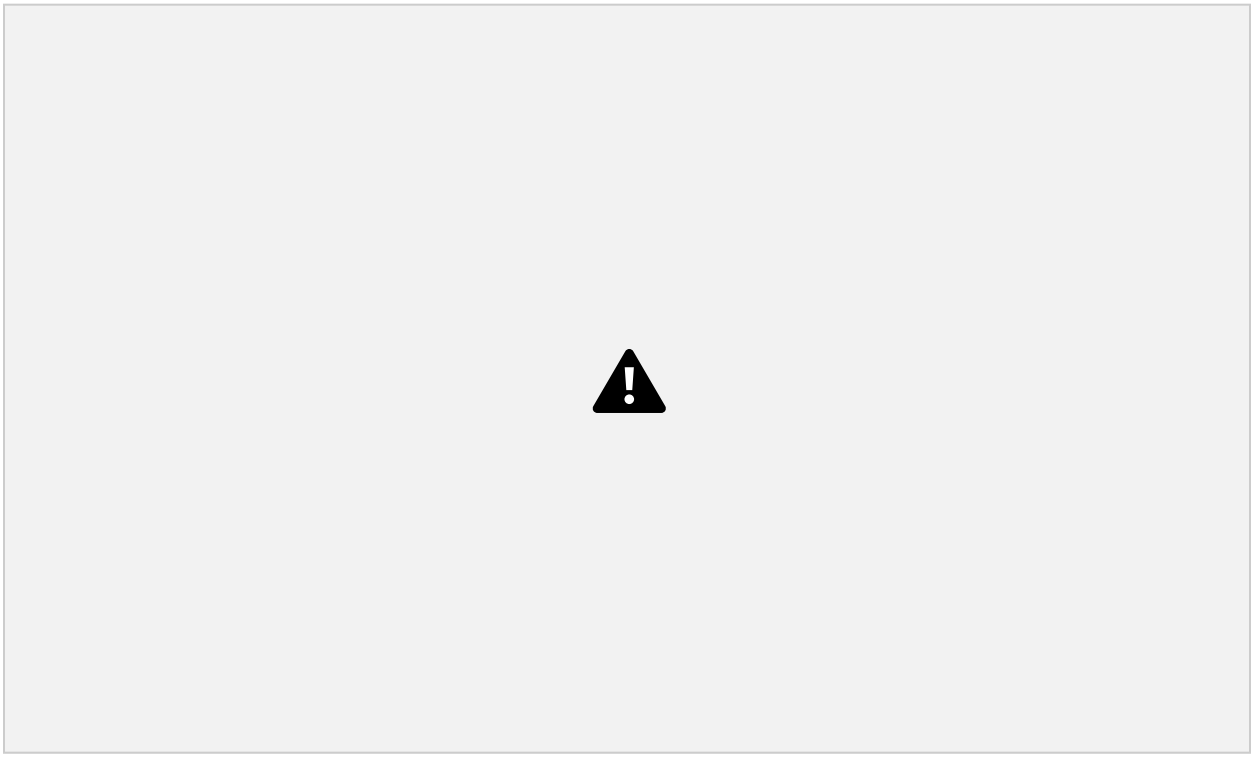


Step 49: Click on Next
Step 50: Choose the created VPC and both the created subnets as well

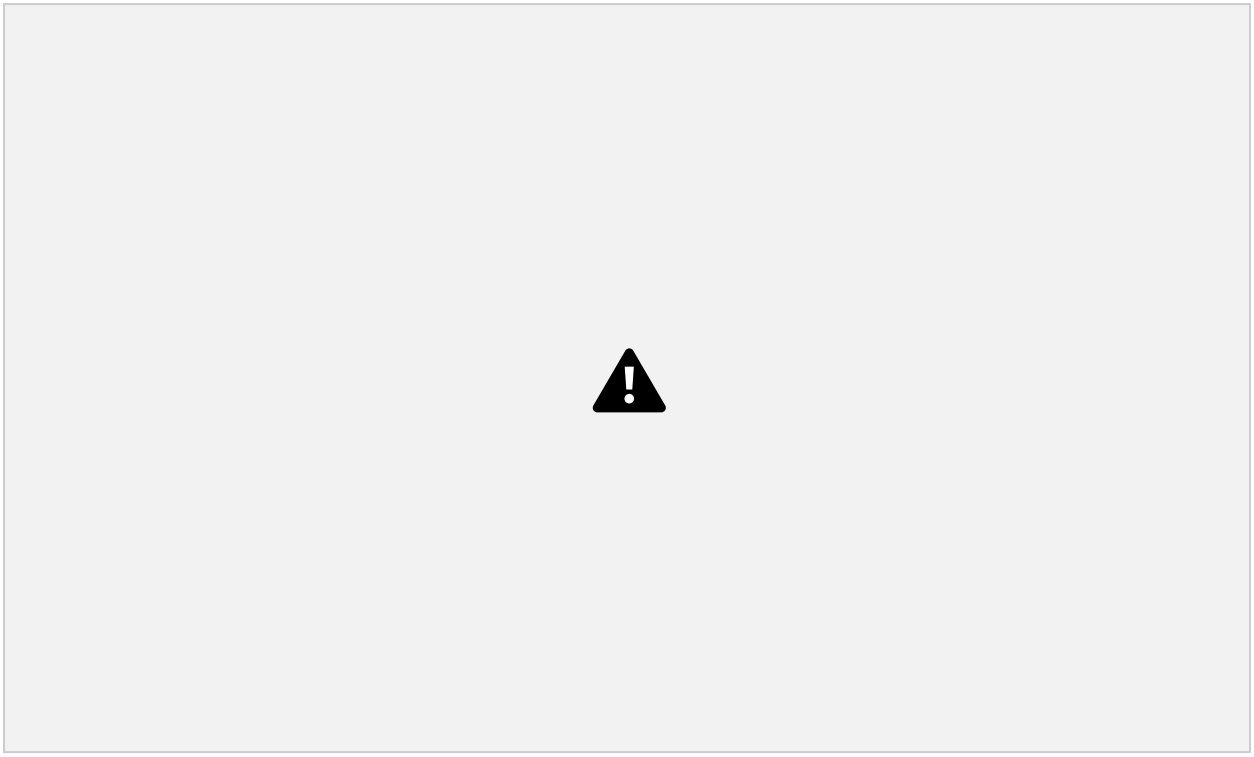


Step 51: Click on Next

Step 52: Select Attach to an Existing Load Balancer and select the created Target group

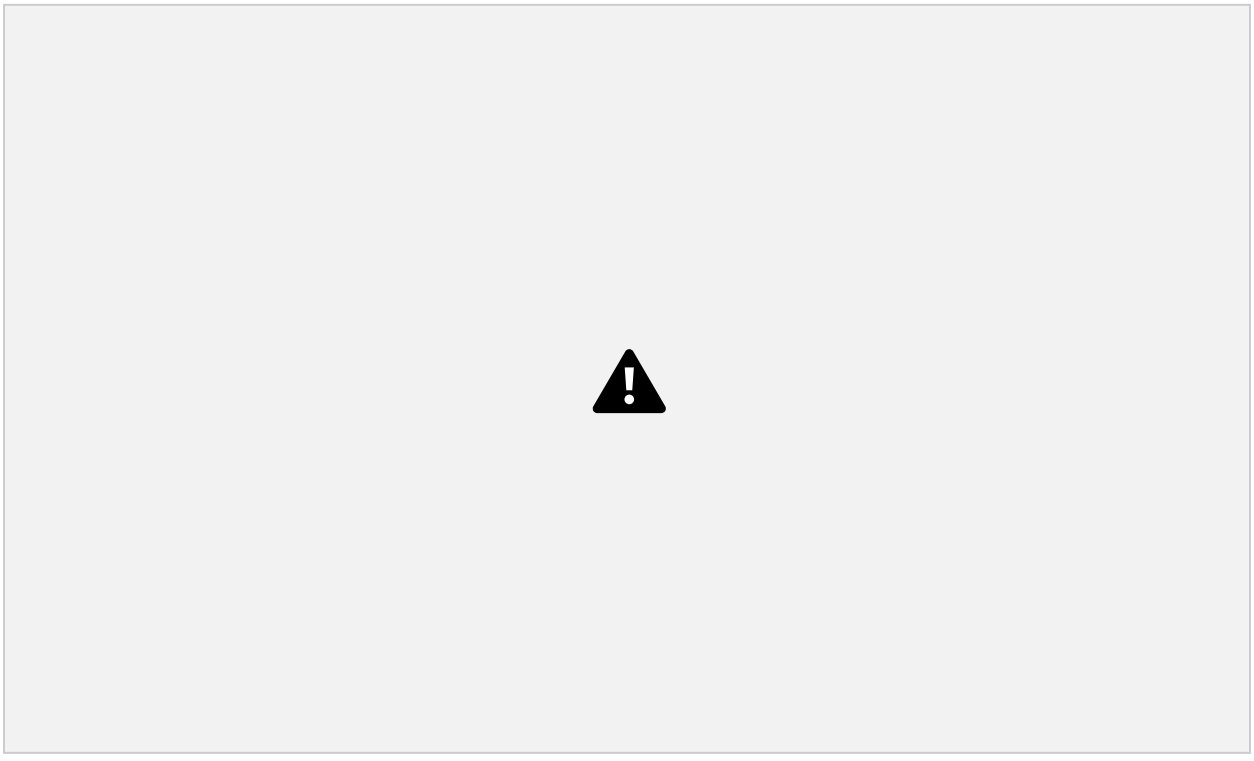


Step 53: Enable “Turn on Elastic Load Balancing health checks”

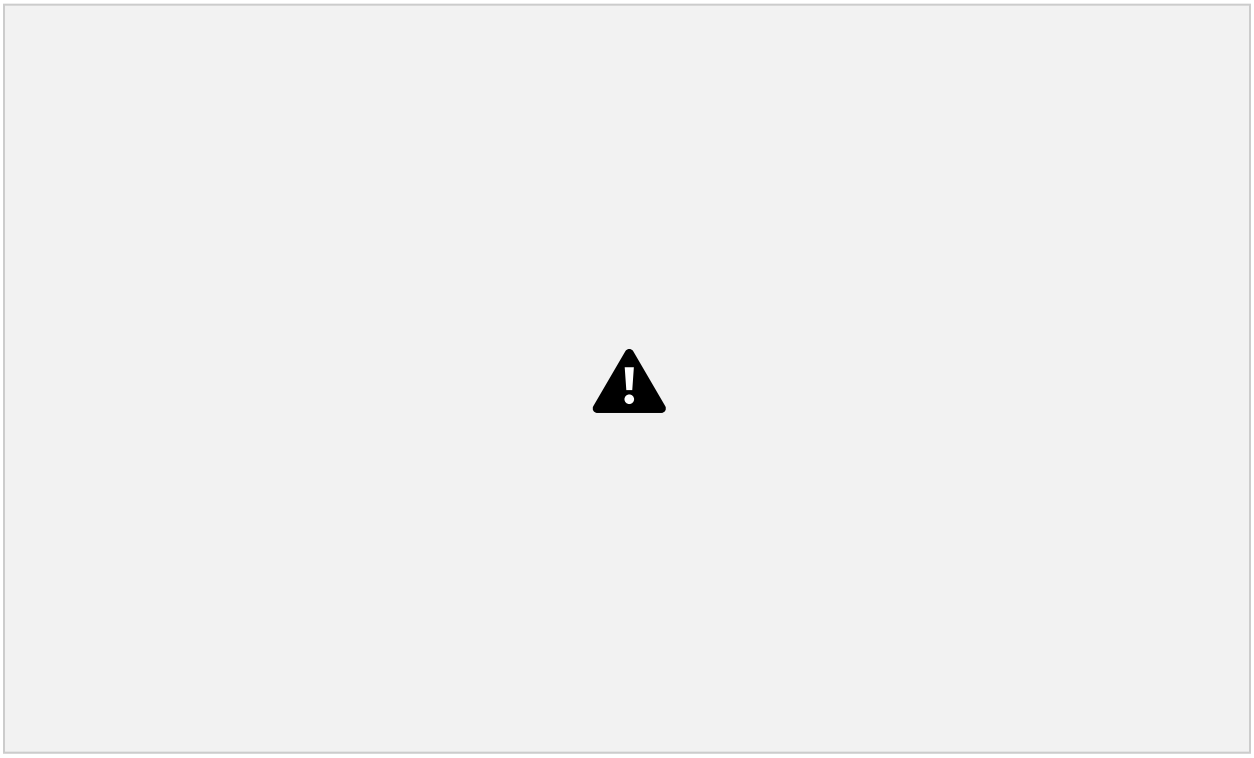


Step 54: Click on Next

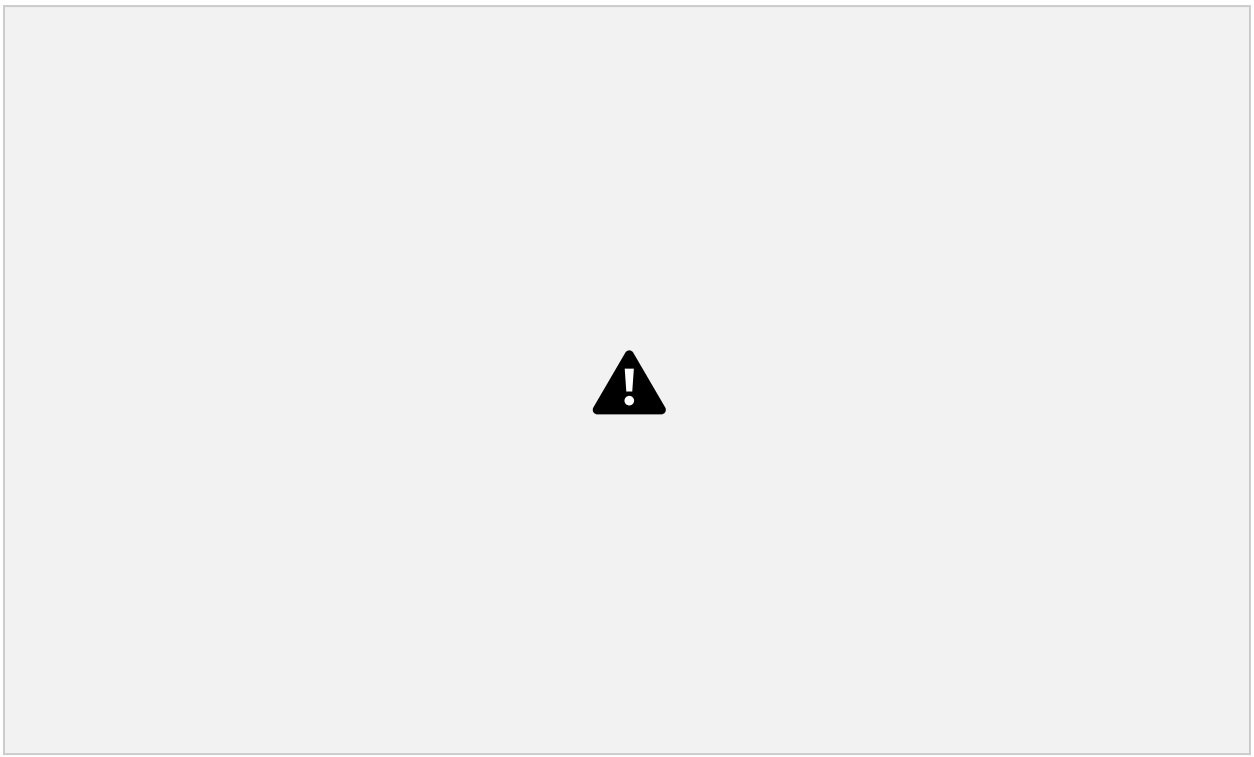
Step 55: Select Desired Capacity: 2
Min Desired Capacity: 1
Max Desired Capacity: 4



Step 56: Click on Next
Step 57: Click on Next
Step 58: Click on Next
Step 59: Click on Create Auto Scaling Group

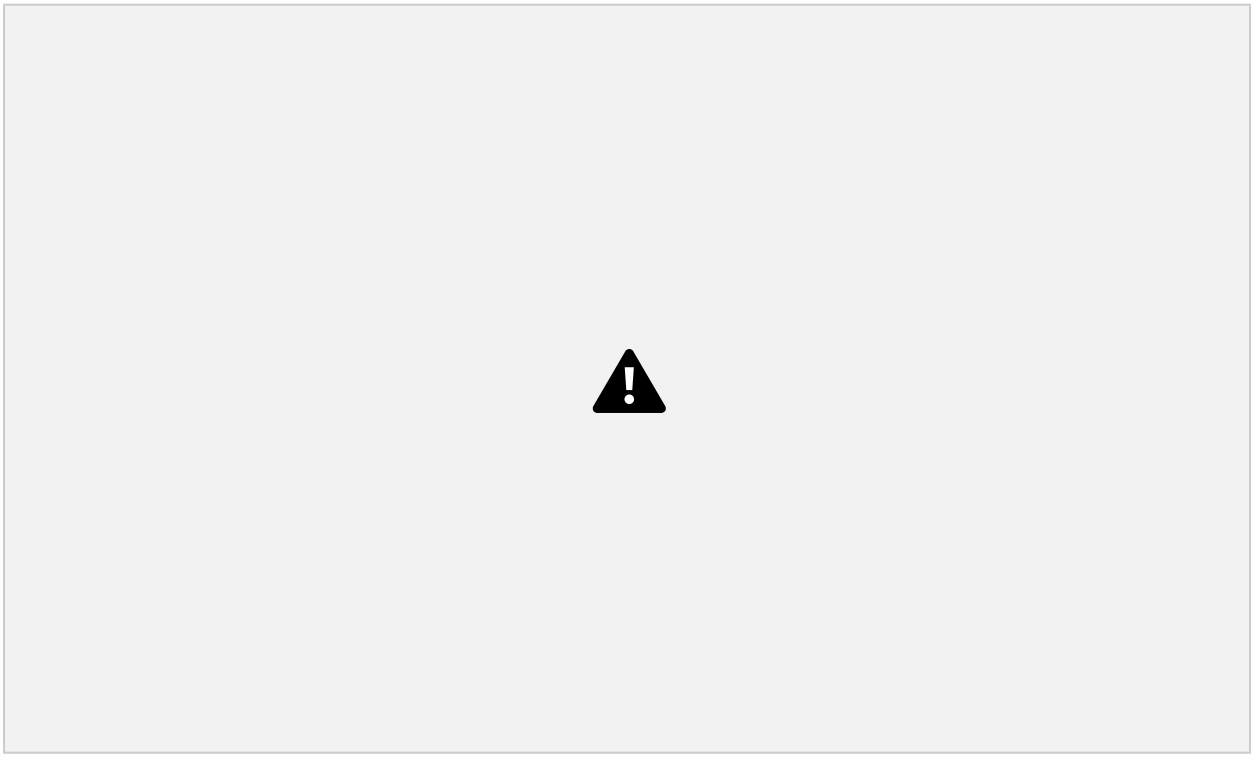


Step 60: Navigate to EC2 and you will 2 instances created by the Auto Scaling Group. Wait for the status check to be cleared



Step 61: Once the Status Check is passed navigate back to load balancer from the left-hand side menu.

Step 62: Copy the DNS Name of the Load Balancer and paste it in a browser.



You will find the Output as the hostname of the two instances created