

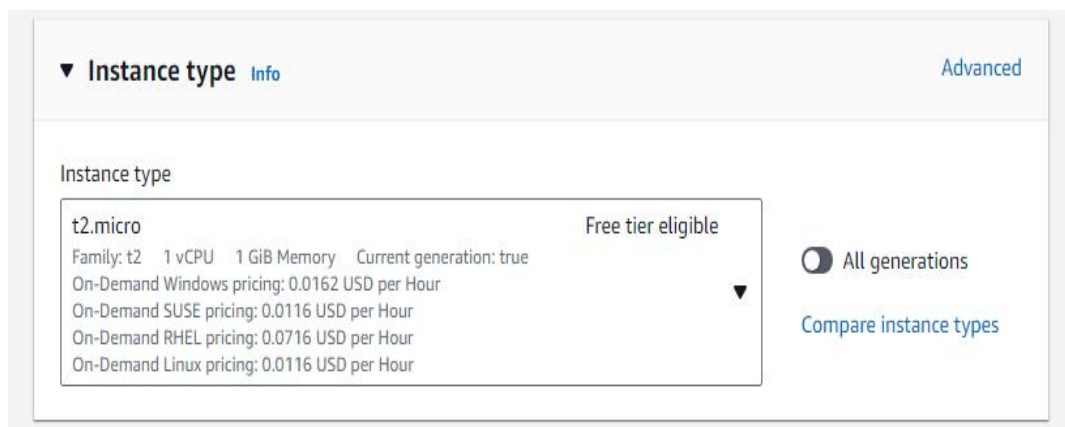
## Implementing High Availability with Auto Scaling and Load Balancer:

Description: Configure an application to automatically scale based on demand to ensure high availability and performance. Implementation: Set up an Auto Scaling group and configure scaling policies based on metrics like CPU utilization or request rate. Use Elastic Load Balancer to distribute traffic across instances.

Setting up High Availability with Auto Scaling involves several steps. Below is a detailed explanation of each step:

### 1. Launch and Ec2 instance.

1. Navigate to the EC2 dashboard.
2. Click "Launch Instance."
3. Choose the Ubuntu Server AMI for both instances.
4. Select an appropriate instance type, like t2.micro.



5. Configure instance details like the number of instances, network settings, etc.
6. Add storage if needed, and configure tags (optional)
7. We'll be using the Amazon Linux 2 AMI and launching instances with a t2.micro instance type.
8. If you already have a key pair, you can proceed with the existing key pair. If not, create a new key pair.
9. For the Security Group, we'll create a new one with rules to allow HTTP/HTTPS and SSH access. This will ensure that our web application can be accessed securely, and we have the necessary SSH access to manage our instances.

## Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
sgr-004feb49e9ab16f44	SSH	TCP	22	Custom <input type="text" value="0.0.0.0/0"/>		Delete
sgr-07a0a2dc98ea8155e	HTTP	TCP	80	Custom <input type="text" value="0.0.0.0/0"/>		Delete
sgr-007fbeb3314ca7a2b	HTTPS	TCP	443	Custom <input type="text" value="0.0.0.0/0"/>		Delete

10. Under the **Advanced Details** section paste this script into the **user data**

### User data - optional [Info](#)

Upload a file with your user data or enter it in the field.

```
#!/bin/bash
# Use this for your user data (script from top to bottom)
# install httpd (Linux 2 version)
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello World from $(hostname -f)</h1>" > /var/www/html/index.html
```

### Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, ...[read more](#)  
ami-03f4878755434977f

### Virtual server type (instance type)

t2.micro

### Firewall (security group)

New security group

### Storage (volumes)

1 volume(s) - 8 GiB

[Review commands](#)

11. This user data script will be executed when each instance is launched, ensuring that the necessary packages are updated, the Apache HTTP server is installed, and a basic HTML file is created to serve as the content for our web server.

## 2. Create an elastic Load Balancer:

1. In the AWS Management Console, go to the EC2 service.
2. Under "Load Balancing," select "Load Balancers."
3. Click "Create Load Balancer."

## Create Classic Load Balancer [Info](#)

The Classic Load Balancer distributes incoming application traffic across multiple EC2 instance targets in multiple Availability Zones. This increases the fault tolerance of your applications. Elastic Load Balancing detects unhealthy instances and routes traffic only to healthy instances.

► **How Classic Load Balancers work**

### Basic configuration

**Load balancer name**  
Name must be unique within your AWS account and can't be changed after the load balancer is created.

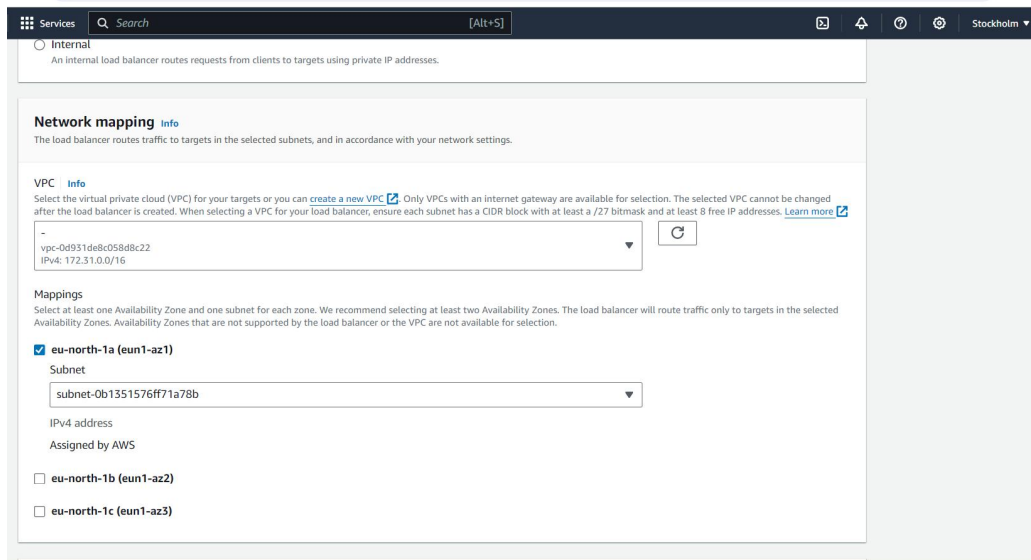
A maximum of 32 alphanumeric characters including hyphens are allowed, but the name must not begin or end with a hyphen.

**Scheme [Info](#)**  
Scheme can't be changed after the load balancer is created.

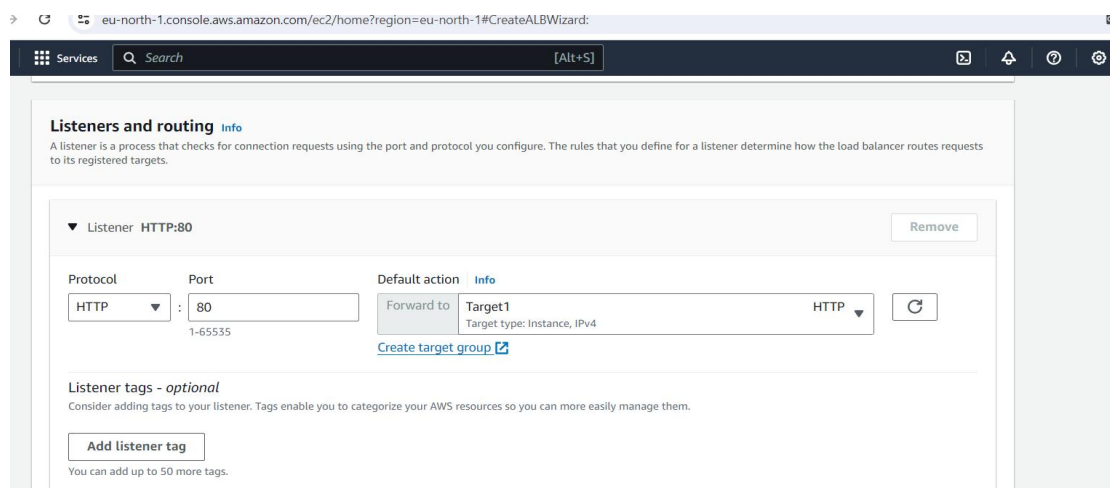
☒ **Internet-facing**  
An internet-facing load balancer routes requests from clients over the internet to targets. Requires a public subnet. [Learn more](#)

☐ **Internal**  
An internal load balancer routes requests from clients to targets using private IP addresses.

4. Choose the appropriate load balancer type.
5. Configure the listener settings, such as port and protocol.
6. Assign the Auto Scaling group instances to the target group.
7. For the 'Target groups,' select the target group that you created earlier. This will specify which instances should receive traffic from the ALB.
8. Ensure that the 'Scheme' is set to 'Internet-facing.' This allows the ALB to receive traffic from the internet.



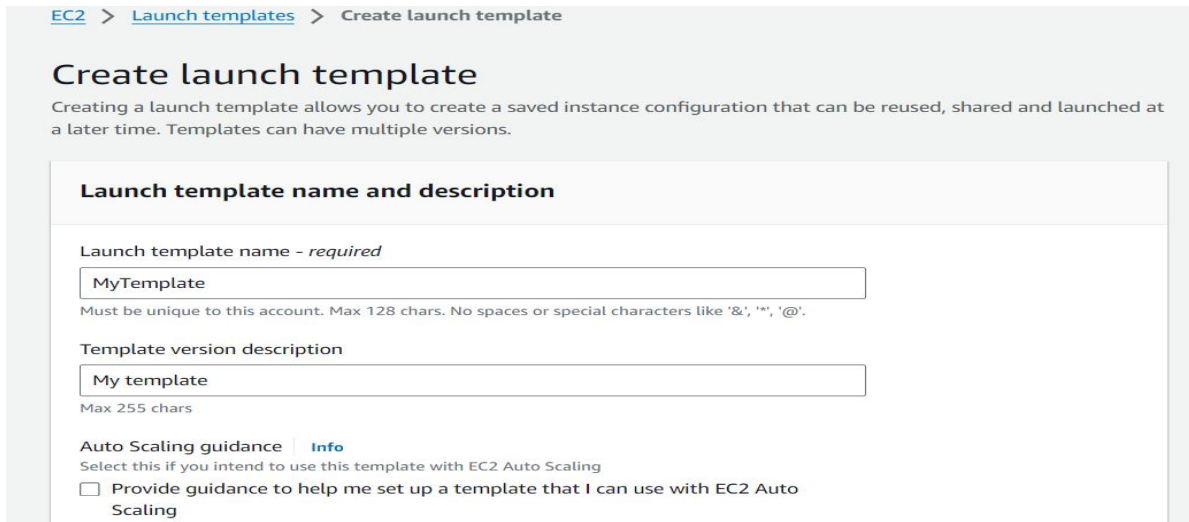
9. Under 'Network mapping,' select the appropriate Virtual Private Cloud (VPC) for your application. Choose the VPC where your instances are located.



10. Select three subnets within the VPC for the ALB to distribute traffic across Availability Zones. This helps improve availability and fault tolerance.
11. For the 'Security groups,' choose the previously created security group that allows HTTP/HTTPS and SSH access. This ensures that the ALB can communicate with the instances securely.

### 3. Create an Auto Saling Group:

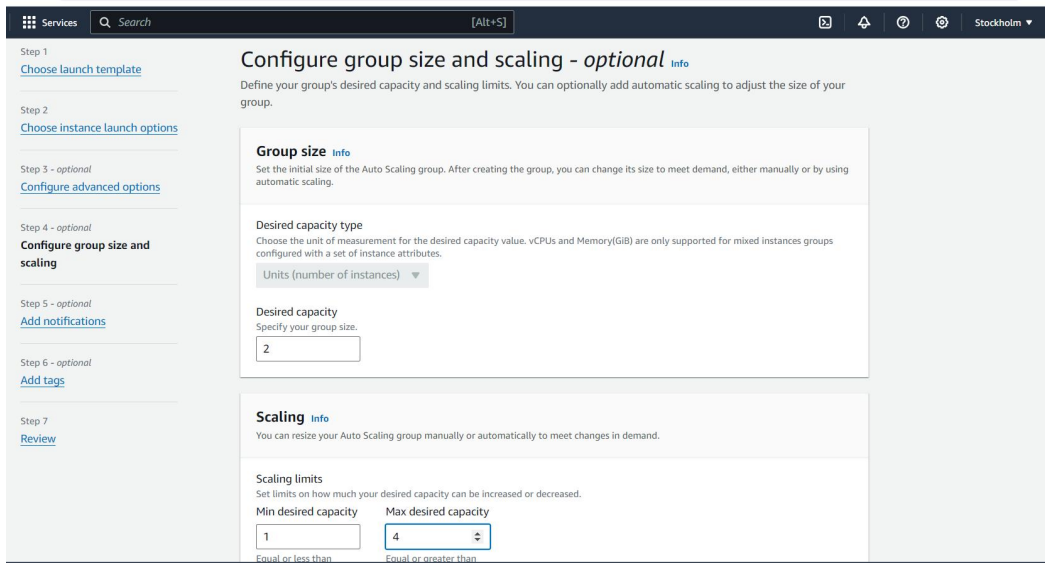
1. Navigate to the AWS Management Console.
2. Open the EC2 service.
3. In the left navigation pane, click on "Auto Scaling Groups."



The screenshot shows the 'Create launch template' page in the AWS Management Console. The breadcrumb navigation at the top reads 'EC2 > Launch templates > Create launch template'. The main heading is 'Create launch template', followed by a subtext: 'Creating a launch template allows you to create a saved instance configuration that can be reused, shared and launched at a later time. Templates can have multiple versions.'

The form is titled 'Launch template name and description'. It contains two text input fields: 'Launch template name - required' with the value 'MyTemplate' and 'Template version description' with the value 'My template'. Below these fields, there is a checkbox labeled 'Auto Scaling guidance' with the text 'Select this if you intend to use this template with EC2 Auto Scaling'. The checkbox is currently unchecked.

4. Click the "Create Auto Scaling group" button.
5. Select Name and Instance Template: Choose a name for your Auto Scaling Group and select the launch template we created earlier.
6. This template includes all the configurations needed for the instances.
7. Select the desired launch configuration or create a new one.



The screenshot shows the 'Configure group size and scaling' page in the AWS Management Console. The breadcrumb navigation at the top reads 'Services > Search > [Alt+S] > Stockholm'. The main heading is 'Configure group size and scaling - optional', followed by a subtext: 'Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.'

The form is titled 'Group size' and contains two sections: 'Desired capacity type' and 'Desired capacity'. The 'Desired capacity type' section has a dropdown menu set to 'Units (number of instances)'. The 'Desired capacity' section has a text input field with the value '2'.

The form is also titled 'Scaling' and contains a section for 'Scaling limits'. This section has two text input fields: 'Min desired capacity' with the value '1' and 'Max desired capacity' with the value '4'.

8. Set up health checks to ensure instances are healthy.

## Health checks

Health checks increase availability by replacing unhealthy instances. When you use multiple health checks, all are evaluated, and if at least one fails, instance replacement occurs.



### EC2 health checks

 Always enabled

### Additional health check types - optional [Info](#)

☒ Turn on Elastic Load Balancing health checks **Recommended**

Elastic Load Balancing monitors whether instances are available to handle requests. When it reports an unhealthy instance, EC2 Auto Scaling can replace it on its next periodic check.

 EC2 Auto Scaling will start to detect and act on health checks performed by Elastic Load Balancing. To avoid unexpected terminations, first verify the settings of these health checks in the Load Balancer console [↗](#) 

☐ Turn on VPC Lattice health checks

VPC Lattice can monitor whether instances are available to handle requests. If it considers a target as failed a health check, EC2 Auto Scaling replaces it after its next periodic check.

### Health check grace period [Info](#)

This time period delays the first health check until your instances finish initializing. It doesn't prevent an instance from terminating when placed into a non-running state.

seconds

9. Choose Instance Scaling Limits: Specify the minimum, desired, and maximum number of instances you want to maintain within the Auto Scaling Group. This allows the ASG to automatically adjust the number of instances based on demand.

## 4. Configure Scaling Policies:

1. In the Auto Scaling group, under the "Automatic scaling" tab, click on "Add policy."
2. Choose a scaling policy type (e.g., Target Tracking Scaling or Step Scaling).
3. Define the scaling policy with appropriate metrics (e.g., CPU utilization, request rate).

### Scaling policies - optional

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. [Info](#)

☒ Target tracking scaling policy  
Choose a desired outcome and leave it to the scaling policy to add and remove capacity as needed to achieve that outcome.

☐ None

#### Scaling policy name

#### Metric type

#### Target value

#### Instances need

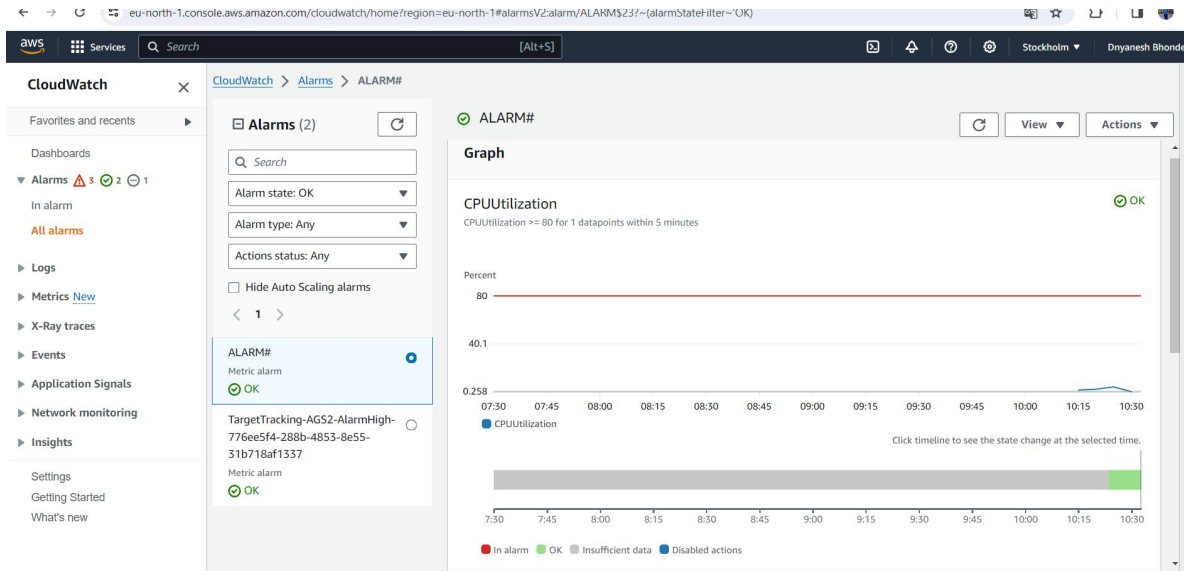
seconds warm up before including in metric

☐ Disable scale in to create only a scale-out policy

4. Set target values or thresholds for the chosen metric.
5. Configure cooldown periods to prevent rapid scaling.

## 5. Test and monitor

1. Deploy your application to the instances launched by the Auto Scaling group.
2. Monitor the Auto Scaling group and Elastic Load Balancer for proper functioning.
3. Simulate load and verify that instances scale in and out based on the defined scaling policies.
4. Ensure traffic is evenly distributed across instances by checking the load balancer.

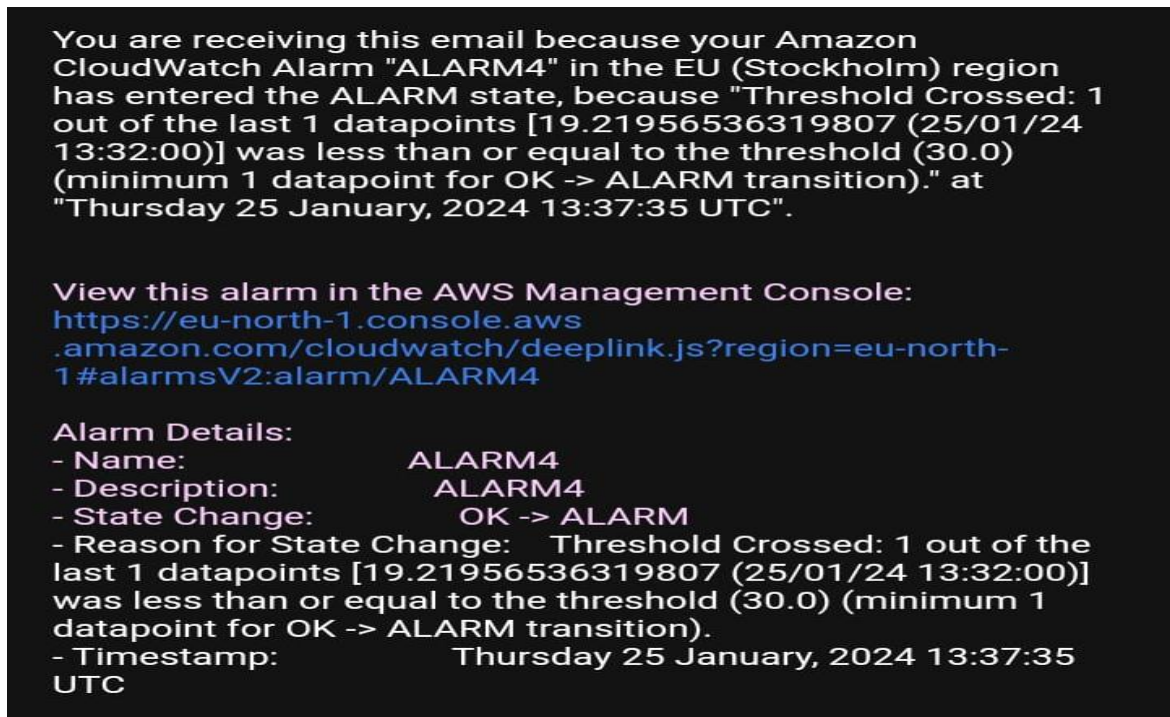


5. You can observe and analyze the performance changes by checking the following AWS monitoring services: CloudWatch, Auto Scaling Group Activity History, and EC2 Instance Monitoring.
6. Finally, cross-check the Application Load Balancer (ALB) DNS to verify whether it successfully distributed the traffic across the instances.





7. Here's a **sample email notification** issued by **CloudWatch** via SNS, however all CloudWatch email alerts look alike.



8. Looking at this email, you can see the **little data available**: name of the alarm triggered, timestamp, region, and state.

## Reference:

<https://docs.aws.amazon.com/whitepapers/latest/real-time-communication-on-aws/high-availability-and-scalability-on-aws.html>

<https://medium.com/@bstuurmanct/achieving-high-availability-with-an-auto-scaling-group-and-an-application-load-balancer-on-aws-8caae425f6c7>

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html>

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/tutorial-ec2-auto-scaling-load-balancer.html>