

# AutoScaleGuard

## Production-Grade Highly Available Web Application with Auto Scaling, Monitoring & Alerting on AWS

**NAME:** NADELLA SRILEKYA

**BATCH:** 06

### Project Overview

**AutoScaleGuard** is a production-grade AWS project that demonstrates how to build a **highly available, self-healing web application** using **Auto Scaling, load balancing, monitoring, and alerting**.

The application automatically:

- Scales out during high CPU load
- Scales in when load decreases
- Monitors system health in real time
- Sends alert notifications to humans

This project follows **real production best practices** and is suitable for **DevOps / Cloud Engineer roles**.

### Project Objectives

- Deploy a **highly available web application** on AWS
- Implement **automatic scaling** based on CPU utilization
- Monitor infrastructure using **CloudWatch**
- Send **email alerts** using SNS
- Validate scaling behavior using **stress testing**
- Visualize system behavior via **CloudWatch Dashboard**

### Why This Project? (Problem It Solves)

#### Problems in Traditional Systems

- Fixed number of servers → cannot handle traffic spikes

- Manual scaling → slow and error-prone
- No visibility → hard to detect performance issues
- No alerts → failures go unnoticed

## What AutoScaleGuard Solves

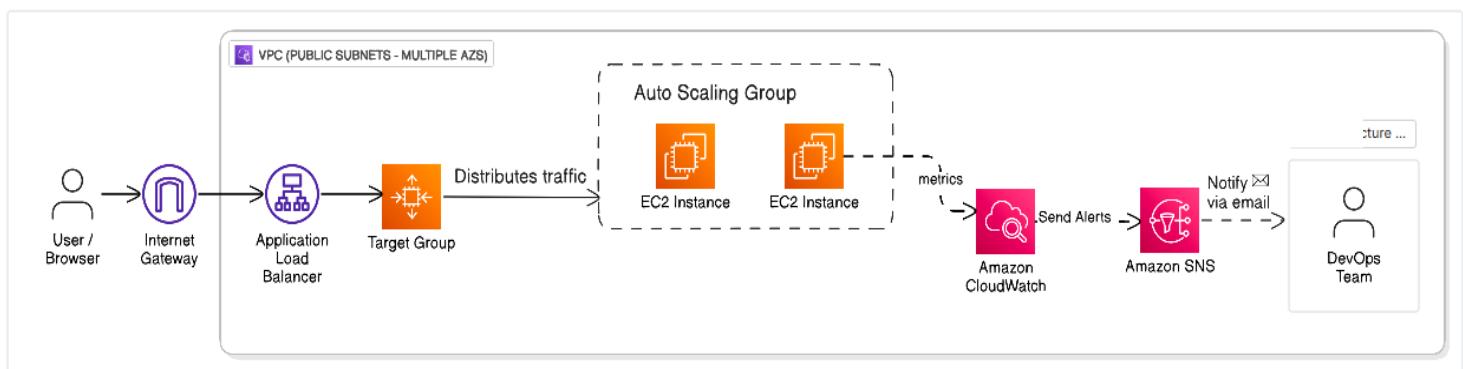
- Handles traffic spikes automatically
- Reduces cost by scaling in when idle
- Provides real-time monitoring
- Sends alerts to engineers
- Ensures high availability across AZs

This is exactly how **modern production systems** are designed.

## System Architecture

### Architecture Pattern:

Stateless web application with Auto Scaling, Load Balancing, Monitoring & Alerting



## Core Components

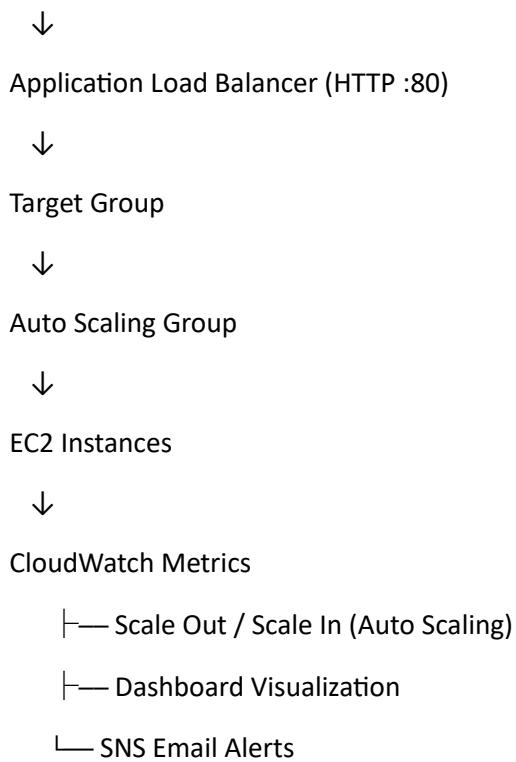
- Application Load Balancer
- Auto Scaling Group
- EC2 instances
- CloudWatch alarms & dashboard
- SNS email notifications

## Request Flow (How the Application Works)

User / Browser



Internet Gateway



### Flow Explanation

1. User sends an HTTP request
2. ALB distributes traffic to healthy EC2 instances
3. EC2 serves the web page
4. CPU metrics are sent to CloudWatch
5. Auto Scaling reacts to CPU thresholds
6. Alerts are sent to email via SNS

## AWS Services Used & Their Purpose

### 1. Amazon EC2

- Hosts the web application
- Uses Amazon Linux 2
- Runs Apache HTTP server
- Stress tool used for testing load

### 2. Application Load Balancer (ALB)

- Distributes traffic across instances
- Ensures high availability
- Health checks unhealthy instances

### **3. Auto Scaling Group (ASG)**

- Maintains desired number of EC2 instances
- Automatically scales based on CPU utilization
- Ensures fault tolerance

### **4. Launch Template**

- Defines EC2 configuration
- Ensures consistency across instances
- Automates application setup via user data

### **5. Amazon CloudWatch**

- Monitors CPU utilization
- Triggers scaling alarms
- Displays metrics via dashboard

### **6. Amazon SNS**

- Sends email notifications
- Alerts humans when thresholds are breached

## **Step-by-Step Implementation**

### **Step 1: Create Security Groups**

Security Groups act as virtual firewalls controlling inbound and outbound traffic to AWS resources.

#### **1.1 Create Application Load Balancer Security Group (alb-sg)**

##### **Purpose:**

Allows public HTTP traffic to reach the Application Load Balancer.

##### **Navigation:**

EC2 → Security Groups → Create Security Group

##### **Configuration:**

- **Name:** alb-sg
- **Description:** Security group for Application Load Balancer
- **VPC:** Default VPC

##### **Inbound Rules:**

Type	Protocol	Port	Source
HTTP	TCP	80	0.0.0.0/0

## Outbound Rules:

- Allow all traffic to 0.0.0.0/0

**Create security group Info**

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

**Basic details**

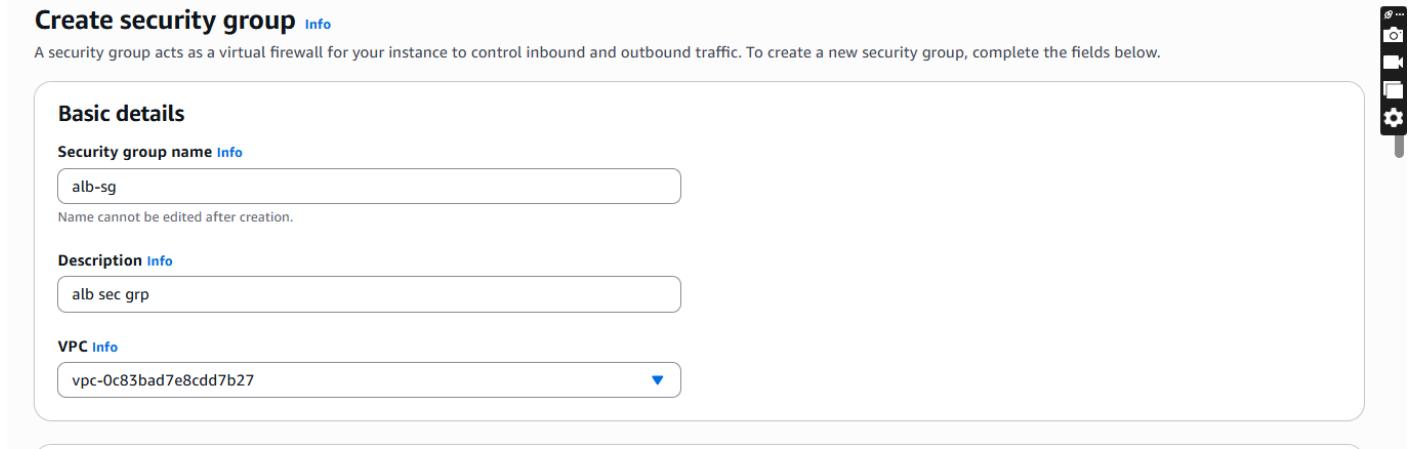
**Security group name Info**  
alb-sg

Name cannot be edited after creation.

**Description Info**  
alb sec grp

**VPC Info**  
vpc-0c83bad7e8cd7b27 ▾

**Inbound rules Info**



## 1.2 Create EC2 Instance Security Group (ec2-sg)

### Purpose:

Restricts direct public access to EC2 instances and allows traffic only from the ALB.

### Navigation:

EC2 → Security Groups → Create Security Group

### Configuration:

- **Name:** ec2-sg
- **Description:** Security group for EC2 instances
- **VPC:** Default VPC

**Create security group Info**

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

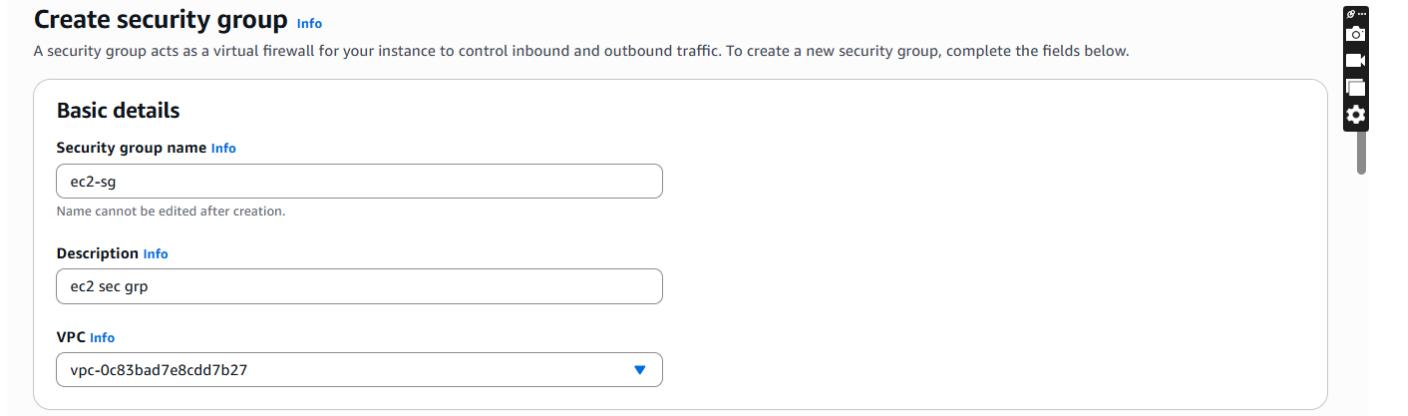
**Basic details**

**Security group name Info**  
ec2-sg

Name cannot be edited after creation.

**Description Info**  
ec2 sec grp

**VPC Info**  
vpc-0c83bad7e8cd7b27 ▾



### Inbound Rules:

Type	Protocol	Port	Source
HTTP	TCP	80	alb-sg
SSH	TCP	22	Your Public IP

### Outbound Rules:

- Allow all traffic

The screenshot shows the AWS Network Firewall inbound rules configuration. There are two rules listed:

- HTTP Rule:** Protocol: TCP, Port range: 80, Source: sg-0389cbc255 (3d old). The source is highlighted with a blue box.
- SSH Rule:** Protocol: TCP, Port range: 22, Source: 49.43.218.22/32. The source is highlighted with a blue box.

At the bottom left is an "Add rule" button.

EC2 instances are **not exposed to the internet** directly — all traffic flows through ALB.

### Step 2: Create Launch Template

A Launch Template defines how EC2 instances are launched in the Auto Scaling Group.

#### Navigation:

EC2 → Launch Templates → Create Launch Template

#### Configuration:

- **Launch template name:** autoscaleguard-lt
- **AMI:** Amazon Linux 2
- **Instance type:** t2.micro
- **Key pair:** Select or create a key pair
- **Security group:** ec2-sg

#### User Data Script

```
#!/bin/bash  
yum update -y  
yum install -y httpd stress
```

```
systemctl start httpd
```

```
systemctl enable httpd
```

```
echo "<h1>AutoScaleGuard Demo</h1>" > /var/www/html/index.html
```

```
echo "<p>Instance ID: $(curl -s http://169.254.169.254/latest/meta-data/instance-id)</p>" >> /var/www/html/index.html
```

This ensures **every EC2 configures itself automatically.**

#### Purpose of User Data:

- Installs Apache web server
- Installs stress tool for load testing
- Automatically starts the web application on boot

### Create launch template

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.

#### Launch template name and description

Launch template name - *required*  
as-lt  
Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '=', '@'.

Template version description  
template  
Max 255 chars

Auto Scaling guidance | Info  
Select this if you intend to use this template with EC2 Auto Scaling  
 Provide guidance to help me set up a template that I can use with EC2 Auto Scaling

#### Summary

Software Image (AMI)

Virtual server type (instance type)

Firewall (security group)

Storage (volumes)

**Create launch template**

Instances

Launch Templates (1/1)

Launch Template ID	Launch Template Name	Default Version	Latest Version	Created
lt-0e673991a61fd21c5	as-lt	1	1	2026-01-

Launch template details

Launch template ID	Launch template name	Default version	Owner
lt-0e673991a61fd21c5	as-lt	1	arn:aws:iam::692809225052:root

## STEP 3: Create Target Group

The Target Group defines where the ALB forwards traffic.

### Navigation:

EC2 → Target Groups → Create Target Group

### Configuration:

- **Target type:** Instance
- **Protocol:** HTTP
- **Port:** 80
- **VPC:** Default VPC

### Health Check Settings:

- **Protocol:** HTTP
- **Path:** /
- **Healthy threshold:** Default

Health checks ensure traffic is routed only to healthy instances.

lb-tg

Target type	Protocol : Port	Protocol version	VPC
Instance	HTTP: 80	HTTP1	vpc-0c83bad7e8cdd7b27

Total targets	Healthy	Unhealthy	Unused	Initial	Draining
0	0	0	0	0	0

## Step 4: Create Application Load Balancer (ALB)

The Application Load Balancer distributes incoming traffic across EC2 instances.

### Navigation:

EC2 → Load Balancers → Create Load Balancer

### Configuration:

- **Load balancer type:** Application Load Balancer
- **Scheme:** Internet-facing

- **IP address type:** IPv4
- **VPC:** Default VPC
- **Subnets:** public-subnet-1, public-subnet-2
- **Security group:** alb-sg

## Listener Configuration

- **Protocol:** HTTP
- **Port:** 80
- **Forward to:** Target Group created earlier

The ALB acts as the **single-entry point** for user traffic.

**Create Application Load Balancer** Info

The Application Load Balancer distributes incoming HTTP and HTTPS traffic across multiple targets such as Amazon EC2 instances, microservices, and containers, based on request attributes. When the load balancer receives a connection request, it evaluates the listener rules in priority order to determine which rule to apply, and if applicable, it selects a target from the target group for the rule action.

▶ How Application 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**  
• Serves internet-facing traffic.

**Internal**  
• Serves internal traffic.

**EC2** < **Load balancers**

**Load balancers (1/1)** What's new?

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	Type	Scheme	IP address type	VPC ID
as-alb	application	Internet-facing	IPv4	vpc-0c8

**Load balancer: as-alb**

**Details** Listeners and rules Network mapping Resource map Security Monitoring

**Details**

Load balancer type	Status	VPC	Load balancer IP address
--------------------	--------	-----	--------------------------

## Step 5: Create Auto Scaling Group (ASG)

The Auto Scaling Group ensures the correct number of EC2 instances are always running.

## Navigation:

EC2 → Auto Scaling Groups → Create Auto Scaling Group

## Configuration:

- **Name:** as-asg
- **Launch template:** as-lt
- **VPC:** Default VPC
- **Subnets:** Same subnets as ALB
- **Attach to load balancer:** Yes
- **Target group:** Previously created target group

## Capacity Settings

Setting	Value
---------	-------

Minimum capacity 1

Desired capacity 1

Maximum capacity 3

The screenshot shows the 'Choose launch template' step of the Auto Scaling group creation wizard. On the left, a sidebar lists steps: Step 1 (Choose launch template) is selected, while Step 2 (Choose instance launch options), Step 3 (optional: Integrate with other services), Step 4 (optional: Configure group size and scaling), Step 5 (optional: Add notifications), Step 6 (optional: Add tags), and Step 7 (Review) are shown as unselected options. The main panel has two sections: 'Name' and 'Launch template'. In the 'Name' section, the 'Auto Scaling group name' field contains 'as-asg', which is described as being unique to the account and Region. In the 'Launch template' section, there is a note: 'For accounts created after May 31, 2023, the EC2 console only supports creating Auto Scaling groups with launch templates. Creating Auto Scaling groups with launch configurations is not recommended but still available via the CLI and API until December 31, 2023.'

## Step 6: Configure Auto Scaling Policy

**Scaling Policy Type:** Target Tracking

## Configuration:

- **Metric:** Average CPU Utilization
- **Target value:** 50%

AWS automatically creates:

- Scale-out alarm
- Scale-in alarm

These alarms **must not be deleted**.

EC2 > Auto Scaling groups > Create Auto Scaling group

**Automatic scaling - optional**  
Choose whether to use a **target tracking policy** | [Info](#)  
You can set up other metric-based scaling policies and scheduled scaling after creating your Auto Scaling group.

No scaling policies  
Your Auto Scaling group will remain at its initial size and will not dynamically resize to meet demand.

Target tracking scaling policy  
Choose a CloudWatch metric and target value and let the scaling policy adjust the desired capacity in proportion to the metric's value.

**Scaling policy name**  
Target Tracking Policy

**Metric type** | [Info](#)  
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

Average CPU utilization

**Target value**  
50

EC2 > Auto Scaling groups

**Auto Scaling groups (1/1)** | [Info](#)  
Last updated less than a minute ago | [Edit](#) | [Launch configurations](#) | [Launch templates](#) | [Actions](#) | [Create Auto Scaling group](#)

Name	Launch template/configuration	Instances	Status	Desired capacity	Min
as-asg	as-lt   Version Default	0	Updating capacity...	1	1

**Auto Scaling group: as-asg**

Details | Integrations | Automatic scaling | Instance management | Instance refresh | Activity | Monitoring | Tags - moved

**as-asg Capacity overview**

arn:aws:autoscaling:us-east-1:1692809225052:autoScalingGroup:36b8e5a5-31b4-4165-bde4-3c31d15f99f0:autoScalingGroupName/as-asg

Scale-out and Scale-in alarms, which are automatically created by AWS by configuring Auto Scaling Policy

CloudWatch > Alarms

**CloudWatch** | [Favorites and recents](#) | [Dashboards](#) | [Alarms](#) (0) | [All alarms](#) | [AI Operations](#) | [GenAI Observability](#) | [Application Signals \(APM\)](#) | [Infrastructure Monitoring](#)

**Alarms (2)**

Hide Auto Scaling alarms | [Clear selection](#) | [Create composite alarm](#) | [Actions](#) | [Create alarm](#)

[Search](#) | [Alarm state: Any](#) | [Alarm type: Any](#)

**Actions status: Any**

Name	State	Last state update (UTC)	Conditions
TargetTracking-asg-AlarmHigh- bd28276c-8536- 4cf7-b78c- 4b81c0ff3041	OK	2026-01-07 16:22:52	CPUUtilization > 50 for 3 datapoints minutes
TargetTracking-asg-AlarmLow- 4b21a88a-0790- 4d75-a286	Insufficient data	2026-01-07 16:20:43	CPUUtilization < 45 for 15 datapoint minutes

## Step 7: Create SNS Topic for Alerts

SNS is used to notify DevOps teams via email when alerts are triggered.

### Navigation:

SNS → Topics → Create Topic

### Configuration:

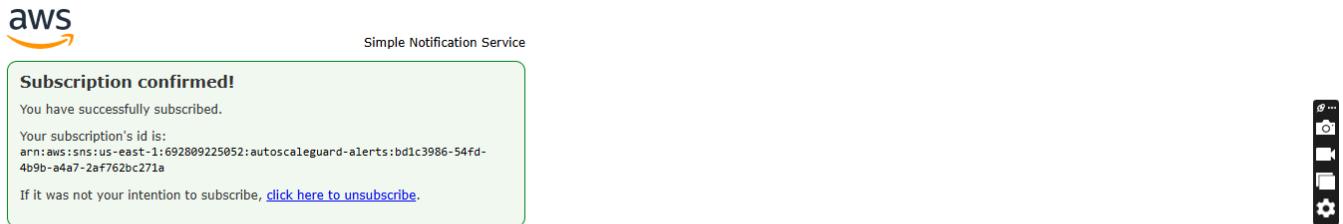
- **Type:** Standard
- **Name:** autoscaleguard-alerts

The screenshot shows the 'Create topic' page in the AWS SNS console. The 'Details' tab is selected. Under 'Type', the 'Standard' option is chosen, highlighted with a blue border. The 'Name' field contains 'autoscaleguard-alerts'. The 'Display name - optional' field is empty. On the right side, there is a vertical toolbar with various icons for managing topics.

## Create Subscription

- **Protocol:** Email
- **Endpoint:** Your email address
- Confirm the subscription from email

The screenshot shows the 'Create subscription' page for the 'autoscaleguard-alerts' topic. The 'Details' tab is selected. The 'Topic ARN' field contains 'arn:aws:sns:us-east-1:692809225052:autoscaleguard-alerts'. The 'Protocol' dropdown is set to 'Email'. The 'Endpoint' field contains 'nsrilekya6336@gmail.com'. A note at the bottom states: 'After your subscription is created, you must confirm it.' with an info icon. The right side features a vertical toolbar.



## Step 8: Create CloudWatch Alarm (Human Alert)

This alarm is for **human awareness**, not auto scaling.

### Navigation:

CloudWatch → Alarms → Create Alarm

### Configuration:

- **Metric:** EC2 → CPUUtilization
- **Statistic:** Average
- **Threshold:** > 70%
- **Period:** 1 minute
- **Evaluation periods:** 2

CloudWatch > Alarms > Create alarm

Step 1  
Specify metric and conditions

Step 2  
Configure actions

Step 3  
Add alarm details

Step 4  
Preview and create

**Specify metric and conditions**

**Metric**

**Graph**  
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 1 minute.

Percent

50

25.1

0.215

14:30      15:30      16:30

**Namespace**  
AWS/EC2

**Metric name**  
CPUUtilization

**InstanceId**  
i-0a10f8d022258adcd

**Instance name**  
No name specified

- Action: Notify SNS topic

**Step 2**

**Configure actions**

Step 3  
Add alarm details

Step 4  
Preview and create

**Notification**

**Alarm state trigger**  
Define the alarm state that will trigger this action.

In alarm  
The metric or expression is outside of the defined threshold.

OK  
The metric or expression is within the defined threshold.

Insufficient data  
The alarm has just started or not enough data is available.

**Send a notification to the following SNS topic**  
Define the SNS (Simple Notification Service) topic that will receive the notification.

Select an existing SNS topic

Create new topic

Use topic ARN to notify other accounts

**Send a notification to...**

autoscaleguard-alerts

Only topics belonging to this account are listed here. All persons and applications subscribed to the selected topic will receive notifications.

Email (endpoints)  
nsrilekya6336@gmail.com - View in SNS Console

**CloudWatch**

Favorites and recents

Alarms 1 1 0

In alarm

All alarms

Billing

AI Operations

GenAI Observability

Application Signals (APM) New

Infrastructure Monitoring

**Alarms (3)**

Hide Auto Scaling alarms

Search

<input type="checkbox"/>	Name	State	Last state update (UTC)	Conditions
<input type="checkbox"/>	Alert Alarm	Insufficient data	2026-01-07 16:35:44	CPUUtilization < 30 for 15 datapoints minute
<input type="checkbox"/>	TargetTracking-asg-AlarmLow-4b21a88a-0790-4d75-a286-963cd6572551	In alarm	2026-01-07 16:35:13	CPUUtilization < 35 for 15 datapoints minutes
<input type="checkbox"/>	TargetTracking-asg-AlarmHigh-bd78276c-8526-4	OK	2026-01-07 16:22:52	CPUUtilization > 50 for 3 datapoints

## Step 9: Generate Load (Stress Test)

To validate scaling behavior, generate artificial load.

### Navigation:

EC2 → Instances → Connect → EC2 Instance Connect

**EC2**

Dashboard

EC2 Global View

Events

**Instances (1)**

Find Instance by attribute or tag (case-sensitive)

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status
<input type="checkbox"/>	i-0a10f8d022258adcd	Running	t3.micro	3/3 checks passed	<input type="button" value="View alarms"/>	

### **Command:**

```
stress --cpu 2 --timeout 1000
```

```
[ec2-user@ip-172-31-82-119 ~]$ stress --version  
stress 1.0.7  
[ec2-user@ip-172-31-82-119 ~]$ stress --cpu 2 --timeout 1000  
stress: info: [28244] dispatching hogs: 2 cpu, 0 io, 0 vm, 0 hdd
```

## Step 10: Verify Auto Scaling

## **Check Auto Scaling Activity:**

- EC2 → Auto Scaling Groups → Activity

The screenshot shows the AWS EC2 Instances page. On the left, a sidebar menu is open under the 'Instances' heading, showing options like 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', and 'Reserved Instances'. The main content area has a header 'Instances (1/1) Info' with a 'Last updated less than a minute ago' timestamp. It includes buttons for 'Connect', 'Instance state', 'Actions', and 'Launch instances'. A search bar says 'Find Instance by attribute or tag (case-sensitive)' and a dropdown says 'All states'. Below is a table with one row:

	Instance ID	Instance state	Instance type	Status check	Alarm status	Av.
	i-0a10f8d022258adcd	Running	t3.micro	3/3 checks passed	1 in alarm	us-

At the bottom, there's a modal window for instance i-0a10f8d022258adcd with tabs for 'Description', 'Details', and 'Logs'.

## Expected behavior:

- New EC2 instance launched
- Total instances increase to 2

The screenshot shows the AWS EC2 Instances page. On the left sidebar, under the 'Instances' section, 'Instances' is selected. In the main content area, the 'Instances (1/3)' table shows three entries. The third entry, 'i-0a10f8d022258adcd', is highlighted with a blue border. Below the table, a detailed view for this instance is shown, including its Instance ID, Public and Private IPv4 addresses, and Instance state (Running).

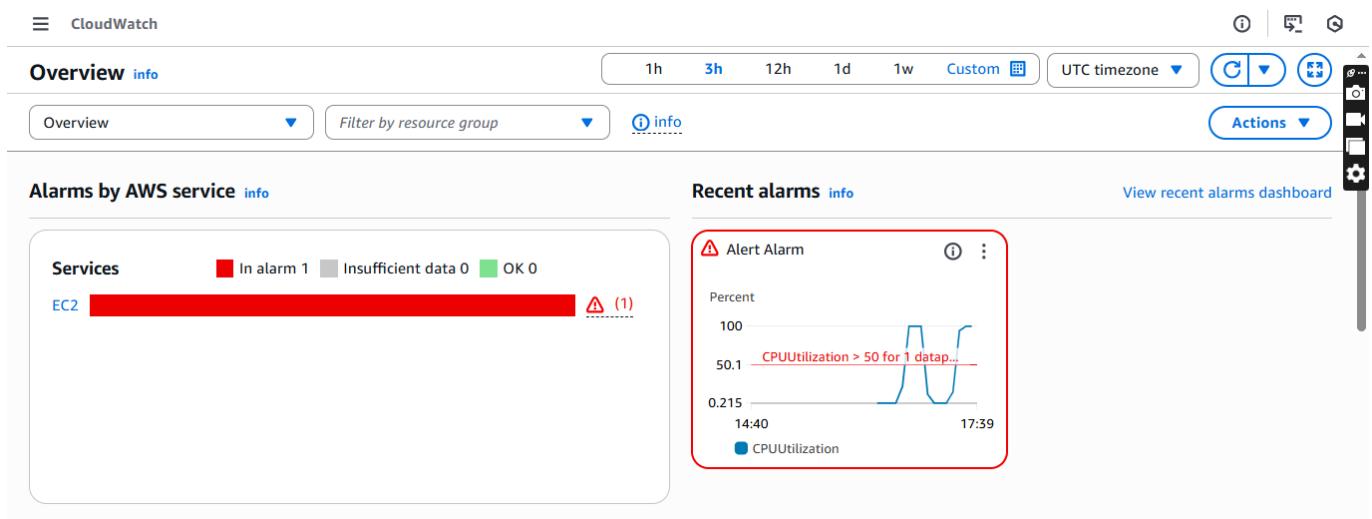
## Alert alarm sends notification to email

The screenshot shows a Gmail inbox. An email from 'AWS Notifications <no-reply@sns.amazonaws.com>' to the user has arrived. The subject of the email is 'ALARM: "Alert Alarm" in US East (N. Virginia)'. The email body contains a message about the CloudWatch Alarm entering the ALARM state due to a threshold being crossed. It includes a link to view the alarm in the AWS Management Console.

## CloudWatch Alarm States

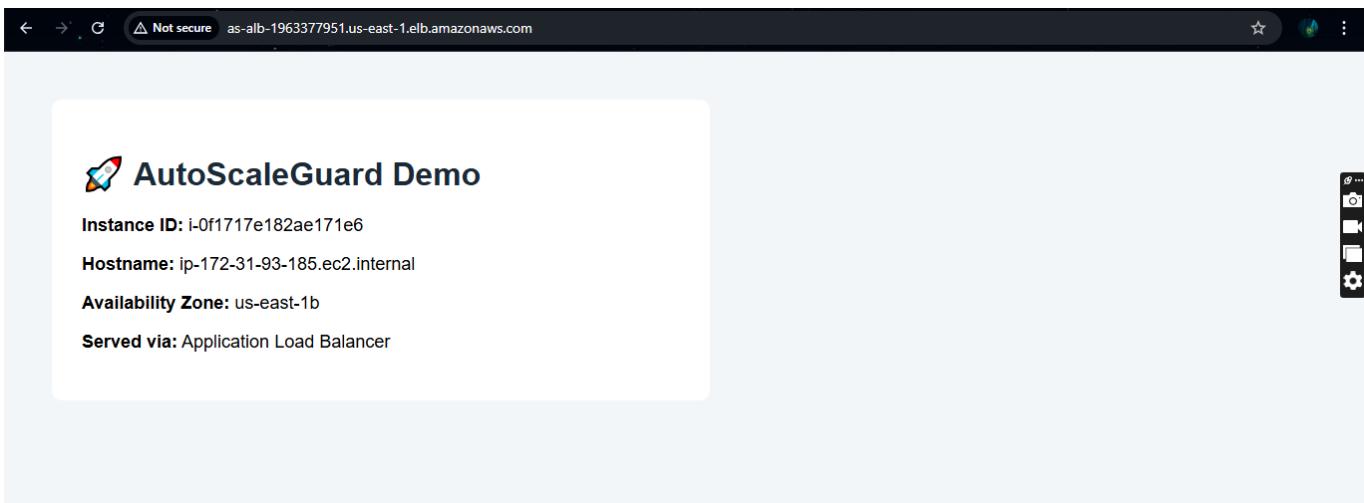
The screenshot shows the CloudWatch Alarms page with the following details:

Name	State	Last state update (UTC)	Conditions
asg-AlarmLow-4b21a88a-0790-4d75-a286-963cd6572551	OK	2026-01-07 16:46:53	CPUUtilization < 35 for 15 datapoints minutes
Alert Alarm	In alarm	2026-01-07 16:46:39	CPUUtilization > 50 for 1 datapoint minute
TargetTracking-asg-AlarmHigh-bd28276c-8536-1234-5678-901234567890	OK	2026-01-07 16:22:52	CPUUtilization > 50 for 3 datapoints minutes



### To verify the deployment:

1. Copy the DNS name of the Application Load Balancer
2. Open it in a web browser



## Step 11: Create CloudWatch Dashboard

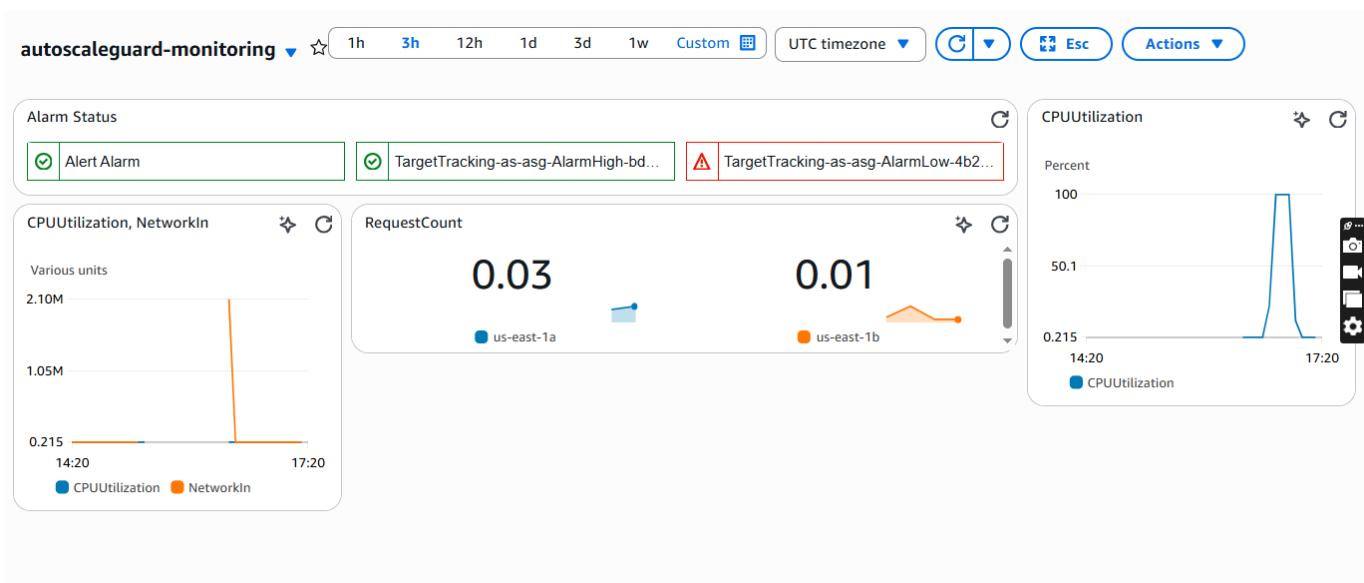
### Navigation:

CloudWatch → Dashboards → Create Dashboard

**Dashboard Name:** autoscaleguard-monitoring

### Widgets

1. Alarm Status (3 alarms)
2. CPU Utilization (per instance)
3. Auto Scaling Metrics:
  - NetworkIn
4. Application Load Balancer Request count



## Conclusion

The AutoScaleGuard project demonstrates the design and implementation of a highly available and scalable web application on AWS using native cloud services. By combining an Application Load Balancer, Auto Scaling Group, and EC2 instances, the application automatically adapts to changing traffic conditions.

Amazon CloudWatch provides real-time monitoring and drives scaling decisions based on CPU utilization, while Amazon SNS ensures that DevOps teams are notified promptly when critical thresholds are breached. The use of launch templates and automated instance configuration enables consistency, reliability, and reduced operational overhead.

Overall, this project reflects real-world production patterns and showcases practical experience in building, monitoring, and operating scalable cloud infrastructure.

## Key Learnings

- Designed a highly available architecture using ALB and Auto Scaling
- Implemented automatic scale-out and scale-in based on CPU utilization
- Gained hands-on experience with CloudWatch metrics, alarms, and dashboards
- Implemented alerting mechanisms using Amazon SNS
- Learned how to validate scaling behavior using stress testing
- Understood the importance of separating scaling alarms from human alerts
- Followed production best practices for security and networking

## Challenges Faced & Troubleshooting

### 1. CloudWatch Alarms Showing “Insufficient Data”

#### Issue:

Alarms initially showed insufficient data.

#### Resolution:

Waited for EC2 metrics to populate and generated CPU load using the stress tool to trigger metric collection.

### 2. Instance ID Not Displaying on Web Page

#### Issue:

Instance metadata was not accessible in the initial user data script.

#### Resolution:

Updated the script to use IMDSv2 tokens to securely retrieve instance metadata.

### 4. No Scale-Out During Load Test

**Issue:**

Scaling did not occur as expected during initial tests.

**Resolution:**

Increased CPU load duration and confirmed scaling policy thresholds and evaluation periods.

**Future Enhancements**

- Enable **HTTPS** using AWS Certificate Manager (ACM)
- Implement **Infrastructure as Code** using Terraform or CloudFormation
- Add **custom application metrics** to CloudWatch
- Integrate alerts with **Slack or PagerDuty**
- Introduce **blue-green or rolling deployments**
- Enhance security using **IAM roles and least-privilege policies**