

# SmartScale: Intelligent EC2 Auto Scaling and Alerting Platform

## Introduction

SmartScale is an enterprise-style cloud automation project that demonstrates **intelligent infrastructure scaling on AWS**. The system continuously monitors EC2 CPU utilization and **automatically launches new EC2 instances when CPU usage exceeds 50%**, ensuring performance stability and high availability. In parallel, the system sends **real-time email notifications** to administrators during scaling events.

The application stack is hosted using **Nginx**, and all new EC2 instances are launched with the **same configuration and output** by using a custom **Amazon Machine Image (AMI)**. This project reflects real-world production scenarios and showcases AWS best practices for monitoring, scaling, and alerting.



## Problem Statement

In traditional server-based systems, sudden traffic spikes can cause performance degradation or downtime due to fixed infrastructure capacity. Manual scaling is slow, error-prone, and inefficient.

This project solves the problem by:

- Automatically detecting high CPU usage
- Launching new EC2 instances without manual intervention
- Notifying administrators instantly via email

## Project Objectives

- Automate EC2 scaling based on CPU utilization
- Maintain identical configuration across all instances
- Implement proactive alerting using email notifications
- Achieve high availability and fault tolerance
- Reduce operational overhead through automation

## Project Overview

- One EC2 instance initially running the application
- Nginx configured to serve the application
- Custom AMI created from the configured EC2 instance
- Launch Template uses the AMI for consistency
- Auto Scaling Group manages instance lifecycle
- CloudWatch monitors CPU utilization
- SNS sends email alerts on scaling events

## AWS Services and Components Used

- **Amazon EC2** – Compute service for hosting application
- **Amazon Machine Image (AMI)** – Preconfigured instance image
- **Launch Template** – Blueprint for EC2 instances
- **Auto Scaling Group (ASG)** – Automatic scaling mechanism
- **Amazon CloudWatch** – Monitoring and alarms
- **Amazon SNS** – Email notification service

- **Nginx** – Web server
- **Amazon VPC** – Network isolation
- **Security Groups** – Controlled inbound and outbound access

## Detailed Architecture Flow

1. A user accesses the application hosted on the EC2 instance through Nginx.
2. Amazon CloudWatch continuously collects CPU metrics from EC2.
3. When CPU utilization exceeds **50%**, CloudWatch triggers an alarm.
4. The alarm invokes the Auto Scaling policy.
5. Auto Scaling Group launches a new EC2 instance using the Launch Template.
6. The Launch Template uses the custom AMI to ensure identical configuration.
7. The new EC2 instance becomes active and starts serving traffic.
8. Simultaneously, Amazon SNS sends an email notification to the user.

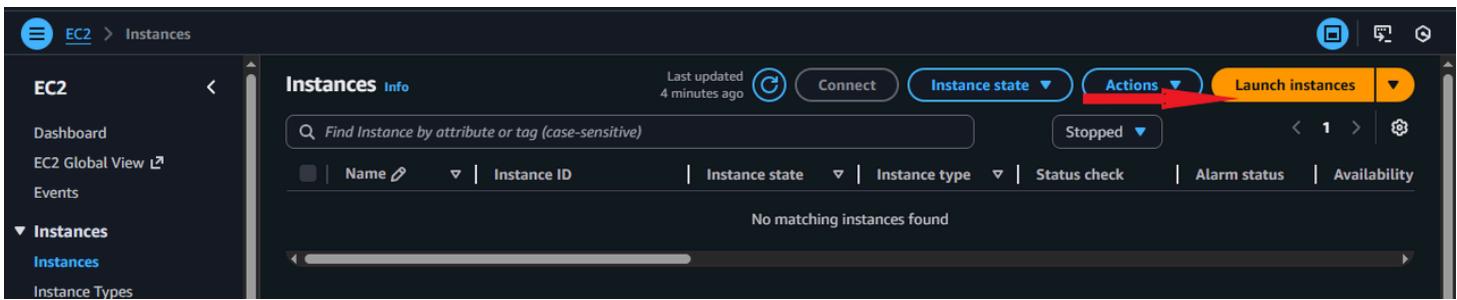
## Auto Scaling Lifecycle

- **Scale-Out Event:** Triggered when CPU > 50%
- **Instance Launch:** New EC2 created using AMI
- **Health Check:** ASG validates instance health
- **Traffic Handling:** New instance serves application
- **Notification:** Email alert sent via SNS

## Step-by-Step Implementation

### Step 1: EC2 Setup and Web Server Configuration

- Launch EC2 using Amazon Linux
- Install and configure Nginx
- Verify application accessibility using public IP



## Step 2: Create Custom AMI

- Select the configured EC2 instance
- Create an AMI to preserve configuration and output
- This AMI ensures uniformity across scaled instances

A screenshot of the 'Create image' page for an EC2 instance. The instance ID is i-08a7507a096e781d9. The 'Image details' section includes:

- Image name:** backup-registration (highlighted with a blue border)
- Image description - optional:** (empty)
- Reboot instance:** checked (with a note about ensuring data consistency)

The 'Instance volumes' section shows a table with columns: Storage type, Device, Snapshot, Size, Volume type, IOPS, Throughput, Delete on termination, and Encrypted. The table currently has one row for an attached volume. At the bottom are links for CloudShell, Feedback, and Console Mobile App, along with copyright and legal information.

## Step 3: Configure Launch Template

- Define AMI, instance type, key pair, and security group
- Launch Template acts as a reusable configuration blueprint

EC2 > Launch templates > Create launch template

### 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

registration-LT<sup>1</sup>

Must be unique to this account. Max 128 chars. No spaces or special characters like '&', '\*', '@'.

Template version description

A prod webserver for MyApp

Max 255 chars

Auto Scaling guidance <sup>Info</sup>

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

▶ Template tags

▶ Source template

#### Launch template contents

Specify the details of your launch template below. Leaving a field blank will result in the field not being included in the launch template.

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

#### Summary

Software Image (AMI)

-

Virtual server type (instance type)

-

Firewall (security group)

-

Storage (volumes)

-

Cancel **Create launch template**

## Step 4: Create Auto Scaling Group

- Attach Launch Template
- Configure:
  - Minimum capacity: 1
  - Desired capacity: 1
  - Maximum capacity: 3
- Enable health checks

AWS Auto Scaling > Scaling plans > Create scaling plan

Step 1 **Find scalable resources** <sup>Info</sup>

Automatically discover or manually choose resources to add to your scaling plan.

Choose a method

Search by CloudFormation stack  
Search for resources provisioned by an AWS CloudFormation stack.

Search by tag  
Search for resources by tags applied to them.

Choose EC2 Auto Scaling groups  
Choose one or more Auto Scaling groups to include in your scaling plan.

Choose Auto Scaling groups <sup>Info</sup>

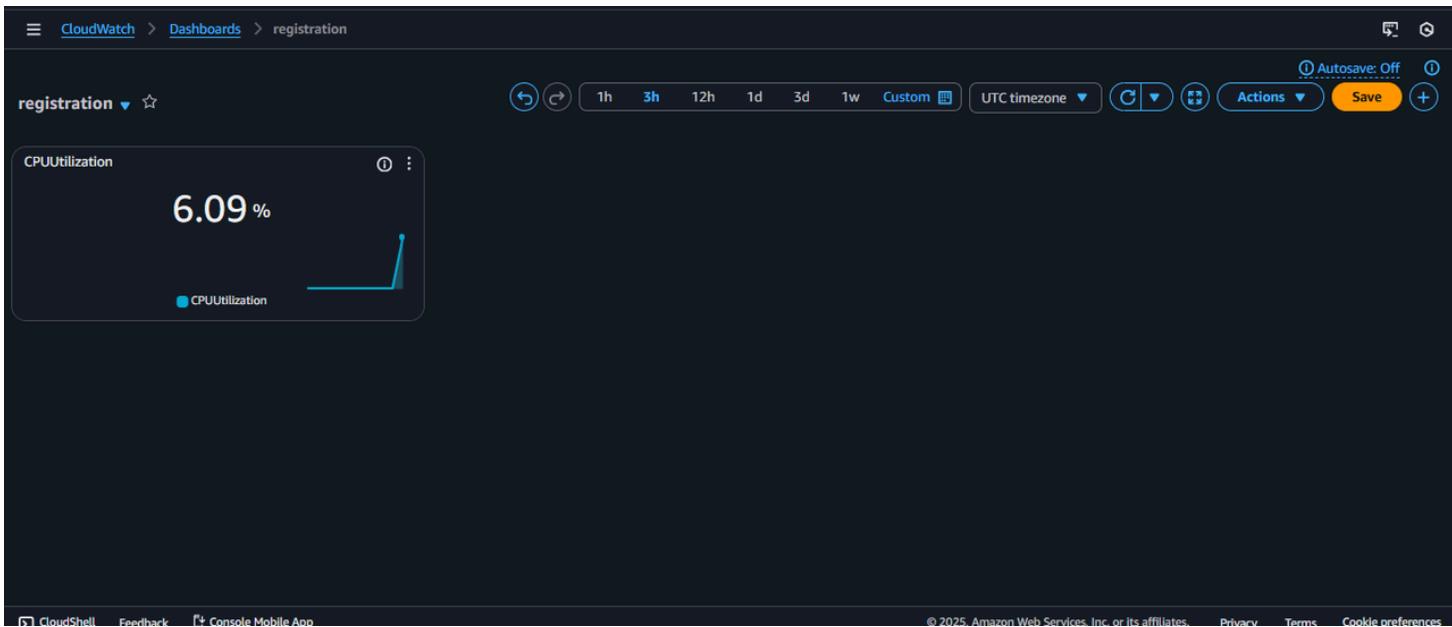
Auto Scaling groups

Choose Auto Scaling groups

**Cancel** **Next**

## Step 5: CloudWatch Monitoring and Alarm

- Monitor CPUUtilization metric
- Create alarm with threshold > 50%
- Configure alarm actions



## Step 6: Scaling Policy Configuration

- Define scale-out policy to add EC2 instance
- Optional scale-in policy to reduce cost

## Step 7: SNS Email Alert Setup

- Create SNS topic
- Subscribe email endpoint
- Confirm subscription
- Attach SNS topic to CloudWatch alarm

Amazon SNS > Topics > Create topic

## Create topic

**Details**

Type | Info Topic type cannot be modified after topic is created

FIFO (first-in, first-out)

- Strictly-preserved message ordering
- Exactly-once message delivery
- Subscription protocols: SQS

Standard

- Best-effort message ordering
- At-least once message delivery
- Subscription protocols: SQS, Lambda, Data Firehose, HTTP, SMS, email, mobile application endpoints

Name  .fifo  
Maximum 256 characters. Can include alphanumeric characters, hyphens (-) and underscores (\_). FIFO topic names must end with ".fifo".

High throughput | Info Configure your FIFO topic for maximum throughput. Once enabled, this configuration cannot be modified.

Message group scope (Recommended for highest throughput)

- Throughput: Maximum regional limits.
- Deduplication: Message deduplication is only verified within a message group.

Topic scope

- Throughput: 3000 message per second and a bandwidth of 20MB per second.
- Deduplication: Message deduplication is verified on the entire FIFO topic.

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Gmail Search mail

Auto Scaling: termination for group "Registration-ASG" Inbox x

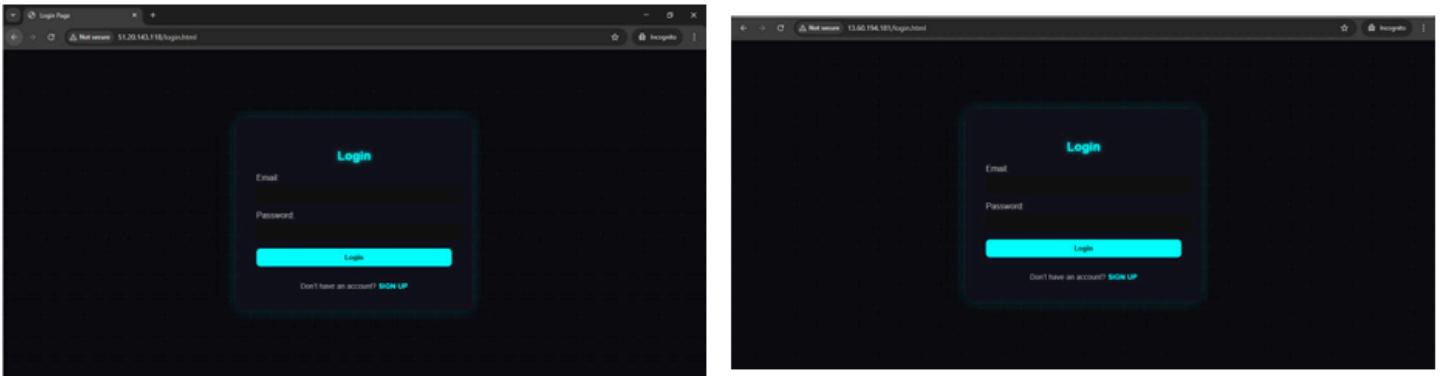
AWS Notifications <no-reply@sns.amazonaws.com> to me 2:56 AM (2 minutes ago)

Service: AWS Auto Scaling  
Time: 2025-12-24T10:56:34.289Z  
RequestId: 7abf40b0-3eb9-4be1-a422-99c124996a66  
Event: autoscaling:EC2\_INSTANCE\_TERMINATE  
AccountId: 586639910260  
AutoScalingGroupName: Registration-ASG  
AutoScalingGroupARN: arn:aws:autoscaling:eu-north-1:586639910260:autoScalingGroup:26206de5-2541-44ac-8c6d-a462d62d8f57:autoScalingGroupName/Registration-ASG  
ActivityId: 7abf40b0-3eb9-4be1-a422-99c124996a66  
Description: Terminating EC2 instance: i-0de29f5431fd10aef  
Cause: At 2025-12-24T10:55:24Z a user request force deleted AutoScaling group changing the desired capacity from 1 to 0. At 2025-12-24T10:55:32Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 1 to 0. At 2025-12-24T10:55:32Z instance i-0de29f5431fd10aef was selected for termination.  
StartTime: 2025-12-24T10:55:32.404Z  
EndTime: 2025-12-24T10:56:34.289Z  
StatusCode: InProgress  
StatusMessage:  
Progress: 50  
EC2InstanceId: i-0de29f5431fd10aef

Reply Forward

## Step 8: Testing and Validation

- Generate artificial CPU load
- Observe automatic EC2 launch
- Verify email notification
- Confirm identical application output



```

[ec2-user@ip-172-31-32-15: ~]
Last login: Mon Dec 24 10:49:38 2015 from 42.104.220.49
[ec2-user@ip-172-31-32-15 ~]$ sudo mysql -u root -p
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 10.5.20-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> use student;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
MariaDB [(none)]> select * from Students;
ERROR 1146 (42000): Table 'TPW.students' doesn't exist
MariaDB [(none)]> select * from TPW.students;
ERROR 1146 (42000): Table 'TPW.students' doesn't exist
MariaDB [(none)]> show tables;
+-----+
| Tables_in_TPW |
+-----+
| student |
+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> select * from student;
+-----+-----+-----+-----+-----+
| id | name | email | password | comment | gender |
+-----+-----+-----+-----+-----+
| 1 | admin | 100000000@gmail.com | pass123 | hi | Male |
| 2 | sally | sally@gmail.com | sally123 | no comments | Female |
| 3 | devendra | dev12@gmail.com | dev123 | | Male |
| 4 | rishabh agar | rishabh12@gmail.com | rishabh123 | | Male |
| 5 | raja shah | raja123@gmail.com | raja123 | hello raja | Male |
| 6 | dev12 | dev12@gmail.com | dev123 | | Male |
| 7 | vishesh | vishesh12@gmail.com | vishesh123 | | Male |
| 8 | riyash | riyash12@gmail.com | riyash123 | hello ryo | Female |
| 9 | valishree | valishree@gmail.com | valishree123 | | Female |
| 10 | right | right12@gmail.com | right123 | | Male |
| 11 | aditya paper | adityap11@gmail.com | adit123 | hello bro | Male |
| 12 | valishree | valishree@gmail.com | valishree123 | | Female |
+-----+
12 rows in set (0.000 sec)

MariaDB [(none)]>

```

```

[ec2-user@ip-172-31-32-15: ~]
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.000 sec)

MariaDB [(none)]> use TPW;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed
MariaDB [(TPW)]> select * from Students;
ERROR 1146 (42000): Table 'TPW.students' doesn't exist
MariaDB [(TPW)]> select * from TPW.students;
ERROR 1146 (42000): Table 'TPW.students' doesn't exist
MariaDB [(TPW)]> show tables;
+-----+
| Tables_in_TPW |
+-----+
| student |
+-----+
1 row in set (0.000 sec)

MariaDB [(TPW)]> select * from student;
+-----+-----+-----+-----+-----+
| id | name | email | password | comment | gender |
+-----+-----+-----+-----+-----+
| 1 | admin | 100000000@gmail.com | pass123 | hi | Male |
| 2 | sally | sally@gmail.com | sally123 | no comments | Female |
| 3 | devendra | dev12@gmail.com | dev123 | | Male |
| 4 | rishabh agar | rishabh12@gmail.com | rishabh123 | | Male |
| 5 | raja shah | raja123@gmail.com | raja123 | hello raja | Male |
| 6 | dev12 | dev12@gmail.com | dev123 | | Male |
| 7 | vishesh | vishesh12@gmail.com | vishesh123 | | Male |
| 8 | riyash | riyash12@gmail.com | riyash123 | hello ryo | Female |
| 9 | valishree | valishree@gmail.com | valishree123 | | Female |
| 10 | right | right12@gmail.com | right123 | | Male |
| 11 | aditya paper | adityap11@gmail.com | adit123 | hello bro | Male |
| 12 | valishree | valishree@gmail.com | valishree123 | | Female |
+-----+
12 rows in set (0.000 sec)

MariaDB [(TPW)]>

```

# Security Considerations

- SSH access restricted via security groups
- HTTP access allowed only on required ports
- IAM roles used for service permissions

# Key Features

- Automated, event-driven scaling
- Zero manual intervention
- Real-time monitoring and alerts
- High availability architecture
- Production-ready design

# Conclusion

SmartScale demonstrates a **robust, scalable, and intelligent cloud infrastructure** capable of adapting to dynamic workloads. By leveraging AWS-native services, the project eliminates manual scaling, improves reliability, and provides proactive alerting. This solution mirrors industry-standard practices and serves as a strong portfolio project for cloud and DevOps roles.

**Project Implemented Using AWS EC2, AMI, Auto Scaling Group, CloudWatch, SNS, and Nginx**