



St. JOSEPH'S
GROUP OF INSTITUTIONS
OMR, CHENNAI - 119



Placement Empowerment Program

Cloud Computing and DevOps Centre

Implement Auto-scaling in the Cloud

Set up an auto-scaling group for your cloud VMs to handle variable workloads.

Name: Monisha JR

Department: CSE



St. JOSEPH'S
COLLEGE OF ENGINEERING



St. JOSEPH'S
INSTITUTE OF TECHNOLOGY

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction :

Auto Scaling in AWS helps manage cloud workloads efficiently by **automatically increasing or decreasing EC2 instances** based on demand. This ensures high availability, cost optimization, and performance efficiency.

In this task, you will:

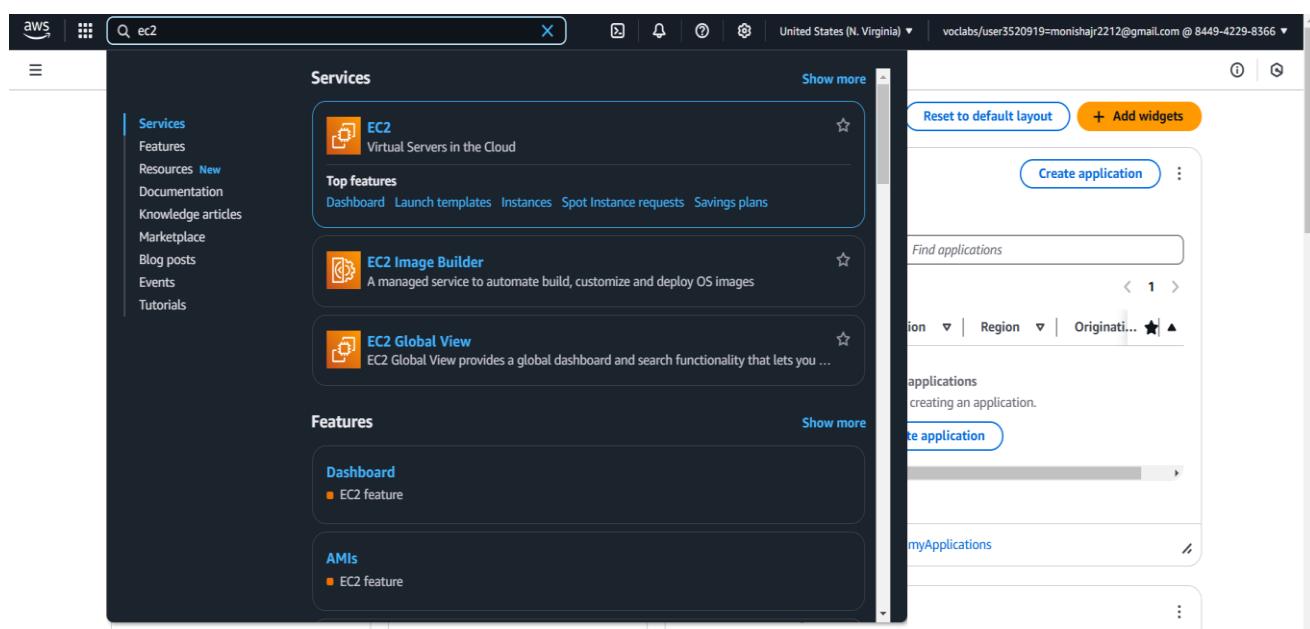
- ✓ Set up an **Auto Scaling Group (ASG)**.
- ✓ Define **scaling policies** to adjust instances based on CPU usage.
- ✓ Simulate high CPU load to **test Auto Scaling**.

Objectives :

1. **Create an EC2 Instance** with a key pair for SSH access.
2. **Set Up an Auto Scaling Group** with a launch template.
3. **Define Scaling Policies** based on CPU utilization.
4. **Simulate High CPU Usage** using the stress tool.
5. **Monitor Scaling Behavior** in the AWS console.

Step-by-Step Overview :

Step 1: Create a Key Pair & Launch an EC2 Instance



The screenshot shows the 'Launch an instance' page in the AWS EC2 console. In the 'Name and tags' section, the 'Name' field contains 'instance1'. There is also a link to 'Add additional tags'.

Launch an EC2 instance (Amazon Linux 2, t2.micro).

The screenshot shows the 'Application and OS Images (Amazon Machine Image)' section. It includes a search bar, tabs for 'Recents', 'My AMIs', and 'Quick Start' (which is selected), and a grid of AMI icons for Amazon Linux, macOS, Ubuntu, Windows, and Red Hat. A button to 'Browse more AMIs' is also present.

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Recents | My AMIs | Quick Start

Amazon Linux | macOS | Ubuntu | Windows | Red Hat

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

The screenshot shows the 'Instance type' section. It lists the 't2.micro' instance type as 'Free tier eligible'. Below this, it provides On-Demand pricing information for various operating systems: Windows, Ubuntu Pro, SUSE, RHEL, and Linux.

t2.micro Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour

On-Demand RHEL base pricing: 0.026 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

All generations

Compare instance types

Create an **SSH key pair** and download the .pem file.

The screenshot shows the 'Create key pair' dialog box over a blurred background of the AWS EC2 Instances page. The dialog has fields for 'Key pair name' (containing 'my_key'), 'Key pair type' (selected 'RSA'), 'Private key file format' (selected '.pem'), and a note about storing the private key securely. At the bottom are 'Cancel' and 'Create key pair' buttons.

Key pair name
my_key

Key pair type
 RSA RSA encrypted private and public key pair

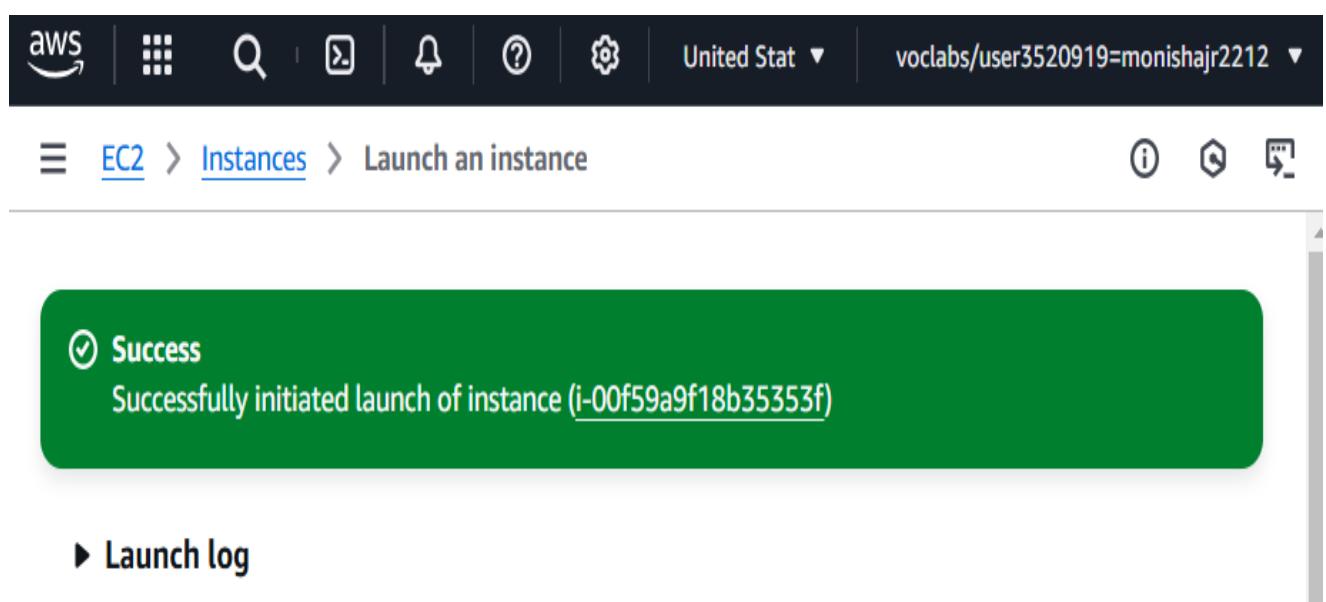
ED25519 ED25519 encrypted private and public key pair

Private key file format
 .pem For use with OpenSSH

.ppk For use with PuTTY

⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

Cancel Create key pair



Step 2 : Set Up an Auto Scaling Group

The screenshot shows the AWS EC2 Auto Scaling homepage. On the left, there's a navigation sidebar with sections like 'Elastic Block Store', 'Network & Security', 'Load Balancing', and 'Auto Scaling'. Under 'Auto Scaling', 'Auto Scaling Groups' is selected. The main content area features a large heading 'Amazon EC2 Auto Scaling' with the subtext 'helps maintain the availability of your applications'. Below this is a brief description of what Auto Scaling groups are. To the right, there's a 'Create Auto Scaling group' button. Further down, there's a 'How it works' diagram showing an 'Auto Scaling group' connected to four EC2 instances. To the right of the diagram is a 'Pricing' section and a 'Getting started' section.

Select a Launch Template (or create a new one).

The screenshot shows the 'Create launch template' wizard. The current step is 'Summary'. It includes fields for 'Software image (AMI)', 'Virtual server type (instance type)', 'Firewall (security group)', and 'Storage (volumes)'. A callout box highlights the 'Free tier' information: 'In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address usage per month, 30 GiB of EBS storage, 2 million I/Os, 1 GiB of snapshots, and 100 GiB of bandwidth to the internet.' At the bottom are 'Cancel' and 'Create launch template' buttons.

The screenshot shows the 'Create security group' wizard. The current step is 'Inbound Security Group Rules'. It lists a single rule: 'Security group rule 1 (TCP, 22, 115.247.189.246/32)'. The rule details are: Type: ssh, Protocol: TCP, Port range: 22, Source type: My IP, Name: 115.247.189.246/32. There are 'Remove' and 'Add' buttons for managing rules.

aws | Search [Alt+S] | United States (N. Virginia) | vclabs/user3520919=monishajr2212@gmail.com @ 8449-4229-8366

☰ EC2 > Launch templates > Create launch template

Success Successfully created mytemp(lt-0ddab40b154c52067).

▶ Actions log

Next Steps

Launch an instance
With On-Demand Instances, you pay for compute capacity by the second (for Linux, with a minimum of 60 seconds) or by the hour (for all other operating systems) with no long-term commitments or upfront payments. Launch an On-Demand Instance from your launch template.

Create instance from this template

Create an Auto Scaling group from your template
Amazon EC2 Auto Scaling helps you maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

Create Auto Scaling group

Create Spot Fleet
A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. The hourly price for a Spot Instance (of each instance type in each Availability Zone) is set by Amazon EC2, and adjusted gradually based on the long-term supply of and demand for Spot Instances. Spot instances are well-suited for data-analysis, batch jobs, background processing, and optional tasks.

Create Spot Fleet

aws | Search [Alt+S] | United States (N. Virginia) | vclabs/user3520919=monishajr2212@gmail.com @ 8449-4229-8366

☰ EC2 > Auto Scaling groups

Auto Scaling groups (1) [Info](#)

Name	Launch template/configuration	Instances	Status	Desired capacity	Min	Max	Availability Zones
auto1	mytemp Version Latest	2	-	2	1	5	us-east-1a, us-east-1b

Step 3: Define Scaling Policies

- Open your Auto Scaling Group.
- Click Automatic Scaling > Add Policy.
- Choose Target Tracking Scaling:
 - Scale up when CPU > 70%.
 - Scale down when CPU < 30%.

Date Created Sat Feb 01 2025 00:12:38 GMT+0530 (India Standard Time)

Details Integrations - new Automatic scaling Instance management Instance refresh Activity Monitoring

Scaling policies resize your Auto Scaling group to meet changes in demand. With reactive dynamic scaling policies, you can track specific CloudWatch metrics and take action when the CloudWatch alarm threshold is met. Use predictive scaling policies along with dynamic scaling policies in the following situations: when your application demand changes quickly, but with a recurring pattern, or when your EC2 instances require more time to initialize.

Dynamic scaling policies (0) [Info](#)

No dynamic scaling policies have been created

Dynamic scaling policies use real-time data to scale your group based on configurable metrics.

aws | ⚡ | 🔍 | 🌐 | 🔔 | ⓘ | ⚙️ | United Stat | voclabs/user3520919=monishajr2212

☰ EC2 > Auto Scaling groups > myauto > Dynamic scaling policy

Create dynamic scaling policy

Policy type

Target tracking scaling

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

70

Instance warmup | Info
300 seconds

Disable scale in to create only a scale-out policy

aws | ⚡ | 🔍 | 🌐 | 🔔 | ⓘ | ⚙️ | United Stat | voclabs/user3520919=monishajr2212

☰ EC2 > Instances > Launch an instance

⌚ Success
Successfully initiated launch of instance (i-00f59a9f18b35353f)

▶ Launch log

Step 4: Simulate High CPU Usage

Connect to the EC2 instance via SSH

```
PS C:\Users\user\Downloads> ssh -i "C:\Users\user\Downloads\my-key.pem" ec2-user@3.86.195.149
The authenticity of host '3.86.195.149 (3.86.195.149)' can't be established.
ED25519 key fingerprint is SHA256:ouCTuzvueMIwJ8S0jSEBmWkmjVROxHVSFn/+JvVUZ2Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.86.195.149' (ED25519) to the list of known hosts.

      _#
     ~\_ #####_      Amazon Linux 2023
    ~~ \_#####\_
    ~~  \###|_
    ~~   \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
    ~~    V~'  __->
    ~~     /_
    ~~-. . /_
    _/_/ _/
    _/m/'

[ec2-user@ip-172-31-81-52 ~]$
```

```
~/n/
[ec2-user@ip-172-31-81-52 ~]$ sudo yum install -y stress
stress --cpu 4 --timeout 300
Last metadata expiration check: 0:06:55 ago on Sat Feb  1 12:04:53 2025.
Dependencies resolved.
=====
 Package           Architecture      Version       Repository      Size
=====
 Installing:
  stress            x86_64          1.0.7-2.amzn2023.0.1      amazonlinux      34 k
 Transaction Summary
=====
 Install 1 Package
 Total download size: 34 k
 Installed size: 68 k
 Downloading Packages:
 stress-1.0.7-2.amzn2023.0.1.x86_64.rpm
=====
 Total
 Running transaction check
 Transaction check succeeded.
 Running transaction test
 Transaction test succeeded.
 Running transaction
  Preparing :                                                 1/1
  Installing : stress-1.0.7-2.amzn2023.0.1.x86_64          1/1
  Running scriptlet: stress-1.0.7-2.amzn2023.0.1.x86_64    1/1
  Verifying  : stress-1.0.7-2.amzn2023.0.1.x86_64          1/1
=====
 Installed:
  stress-1.0.7-2.amzn2023.0.1.x86_64
 Complete!
stress: info: [2986] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

Step 5: Monitor Auto Scaling in AWS Console

- Go to **EC2 Dashboard > Auto Scaling Group.**
- Check if **new instances** are launched when CPU usage increases.
- Verify that **instances terminate** when CPU usage drops.

Outcome :

- Auto Scaling **works correctly** based on CPU usage.
- Instances **scale up** and **scale down** automatically.
- System is **optimized for performance & cost efficiency**.