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:

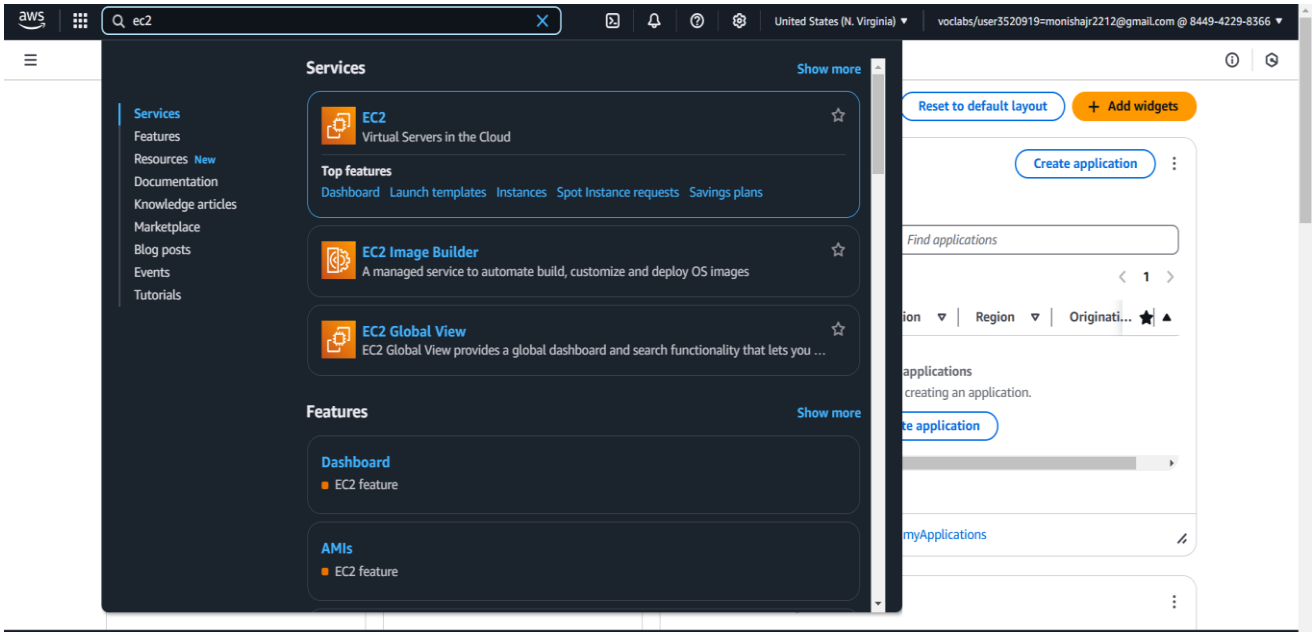
- ✔ Set up an **Auto Scaling Group (ASG)**.
- ✔ Define **scaling policies** to adjust instances based on CPU usage.
- ✔ Simulate high CPU load to **test** Auto Scaling.

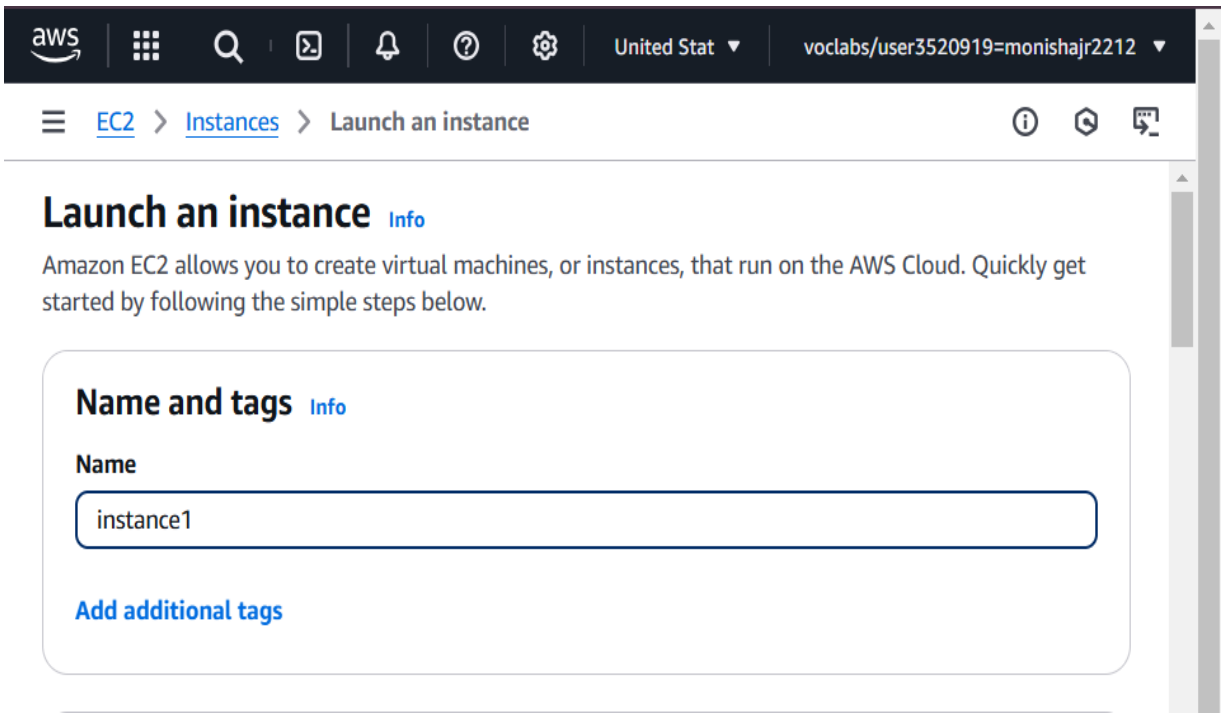
📌 Objectives :

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.

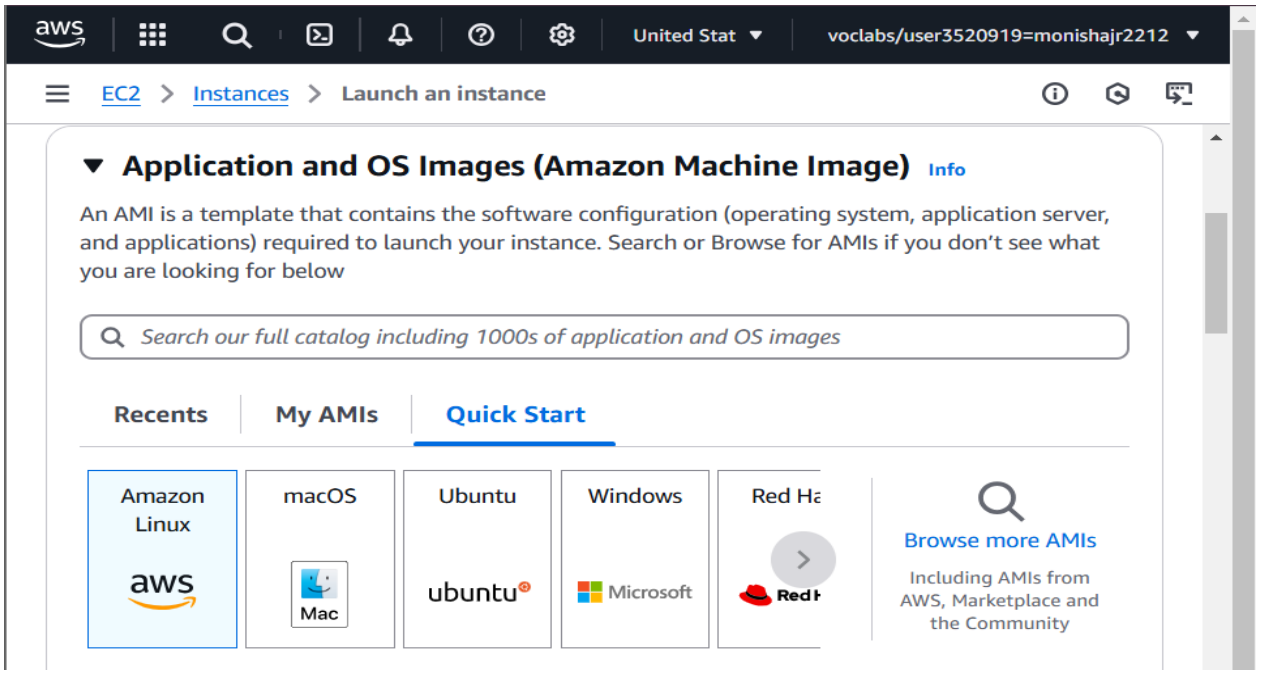
Step-by-Step Overview :

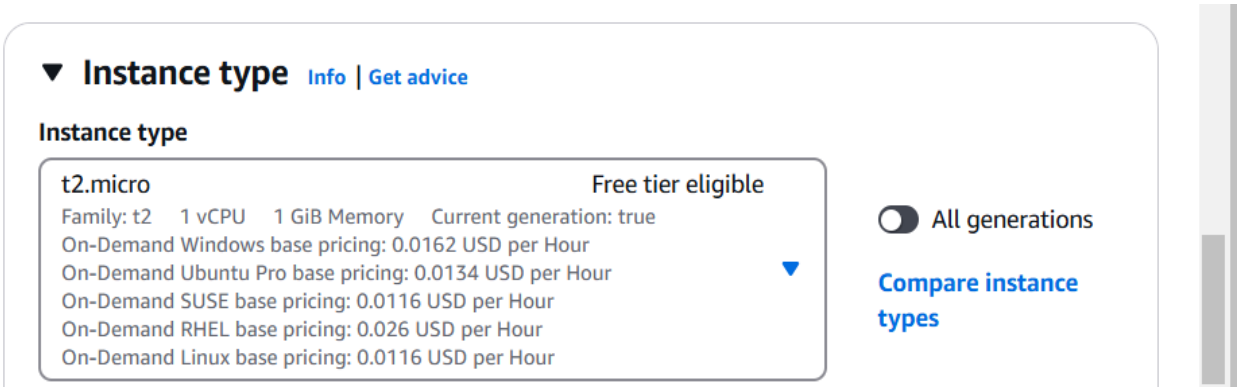
Step 1: Create a Key Pair & Launch an EC2 Instance



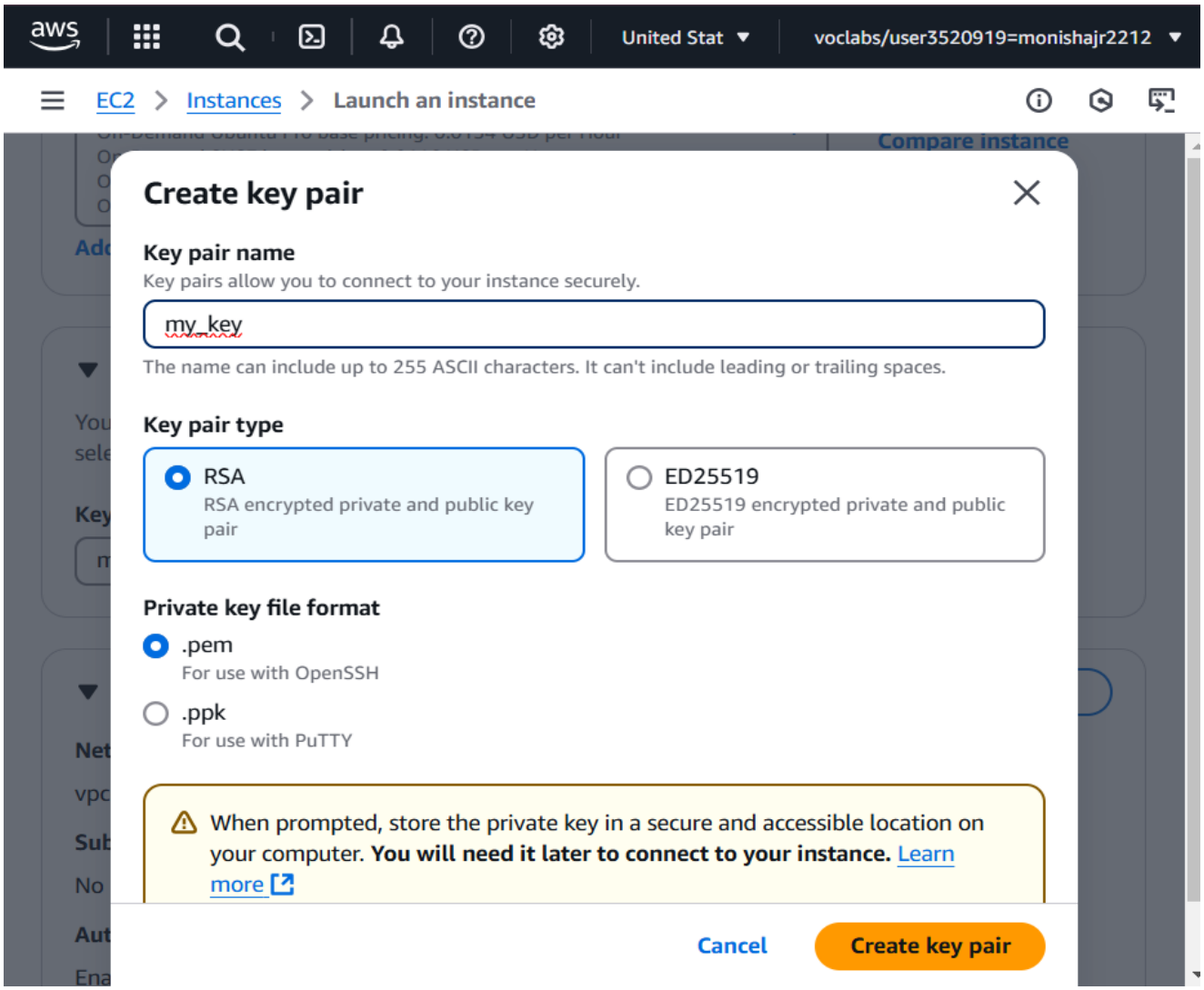


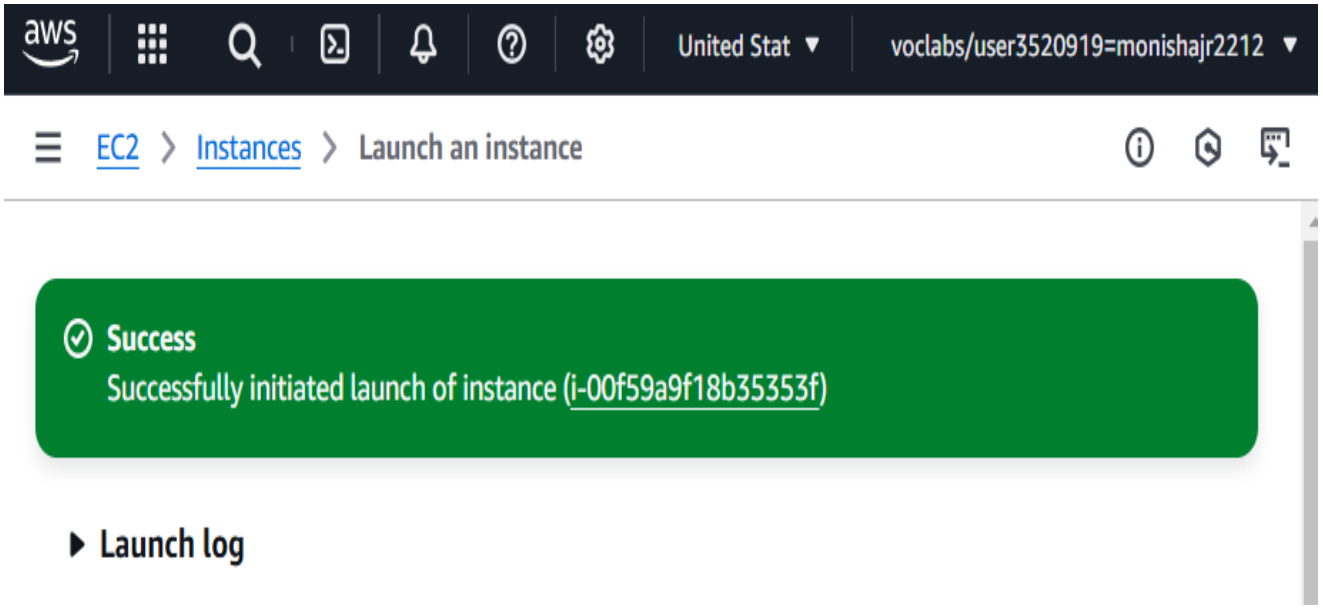
Launch an EC2 instance (Amazon Linux 2, t2.micro).



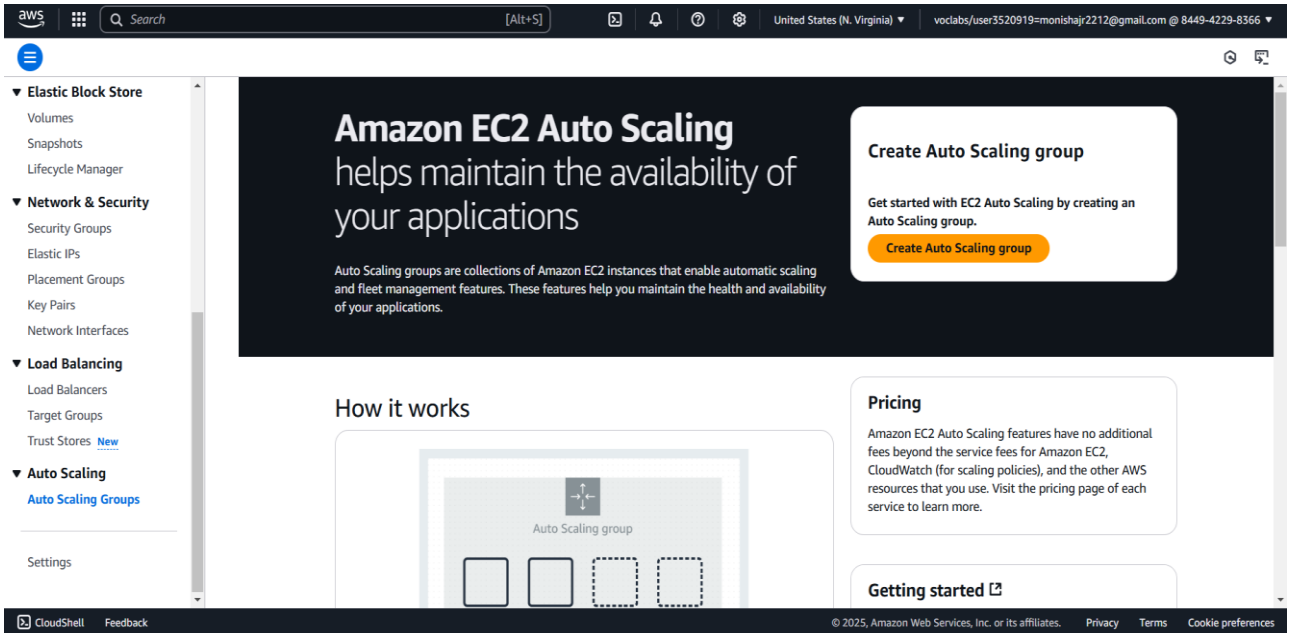


Create an **SSH key pair** and download the .pem file.

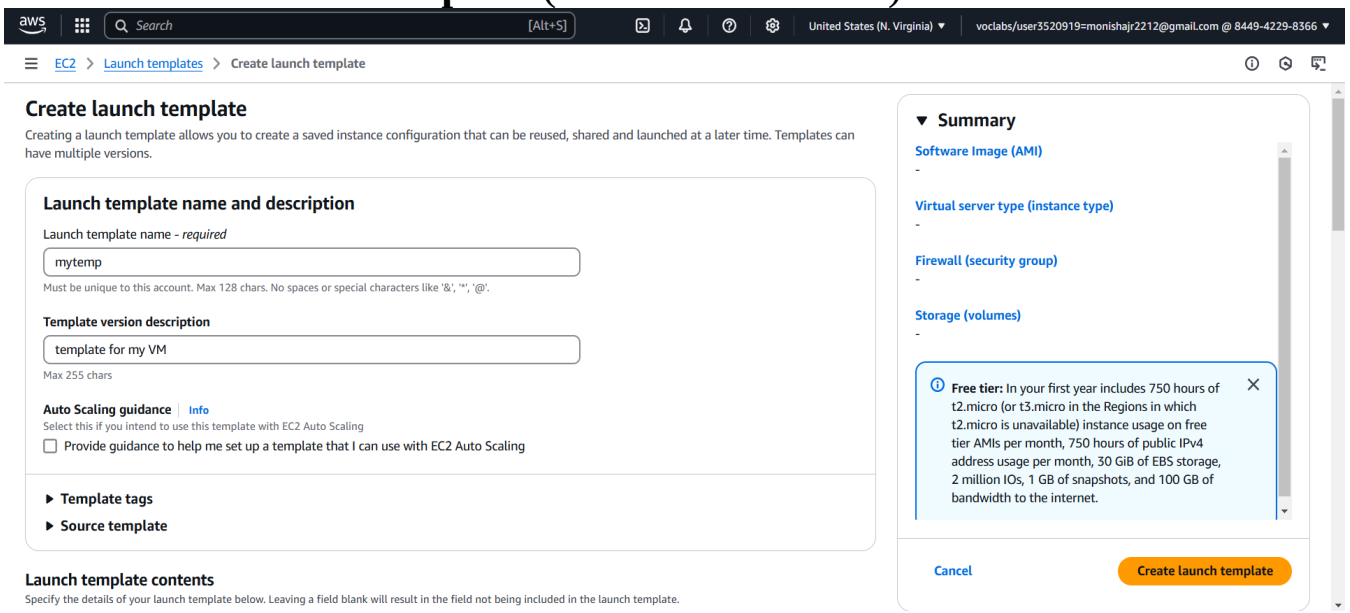


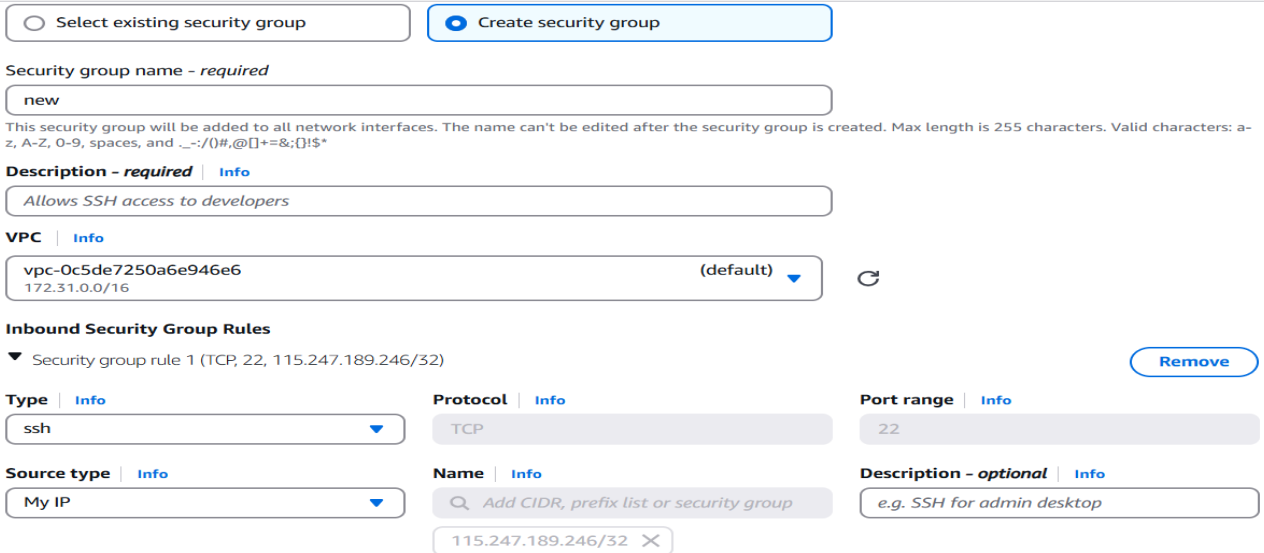


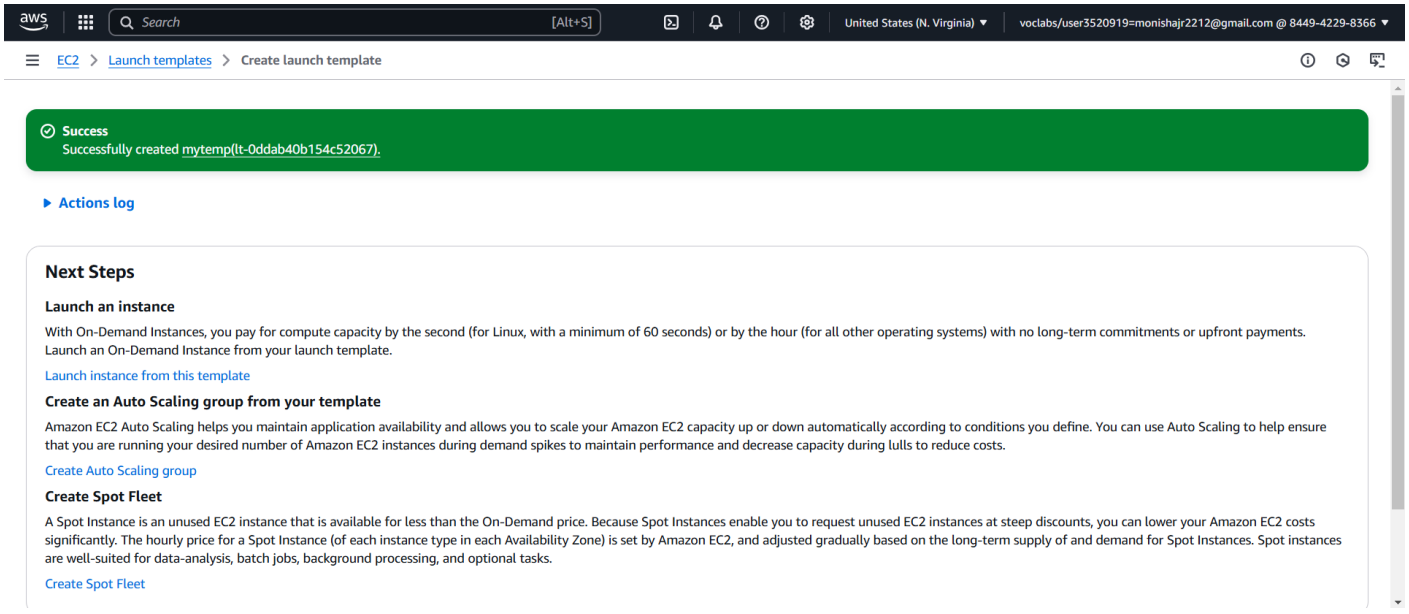
Step 2 : Set Up an Auto Scaling Group

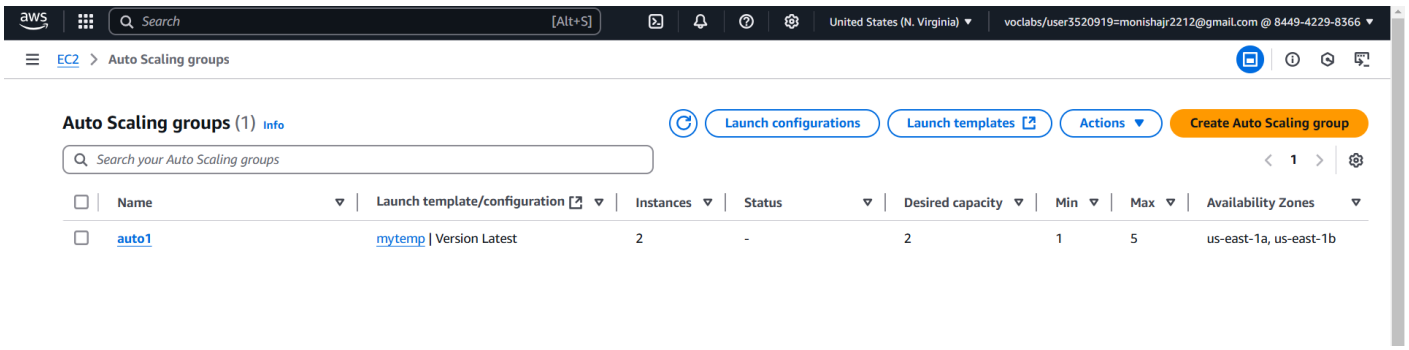


Select a **Launch Template** (or create a new one).



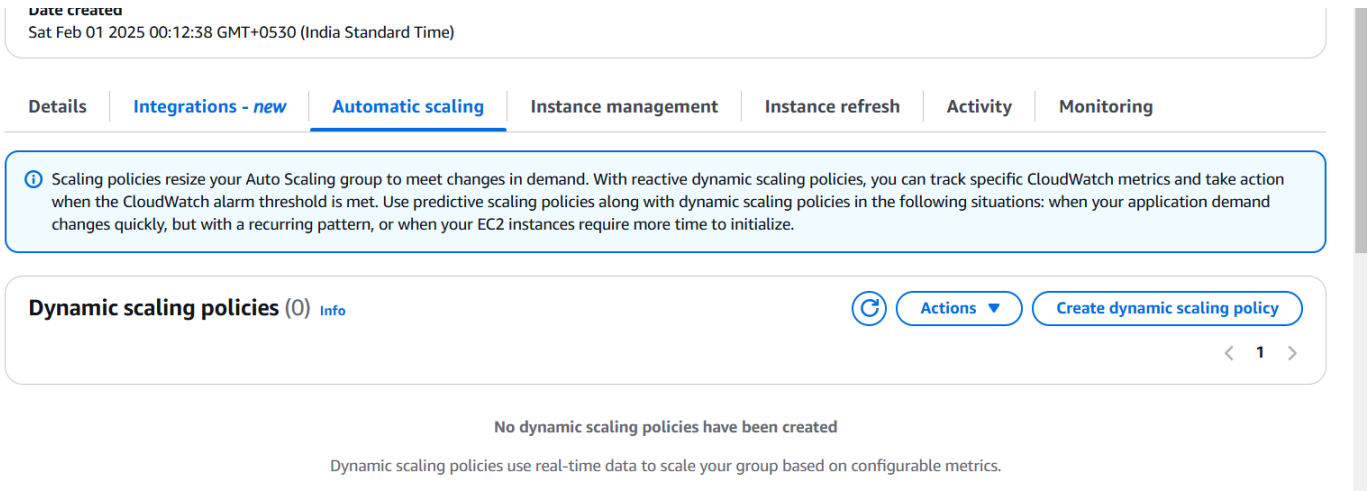


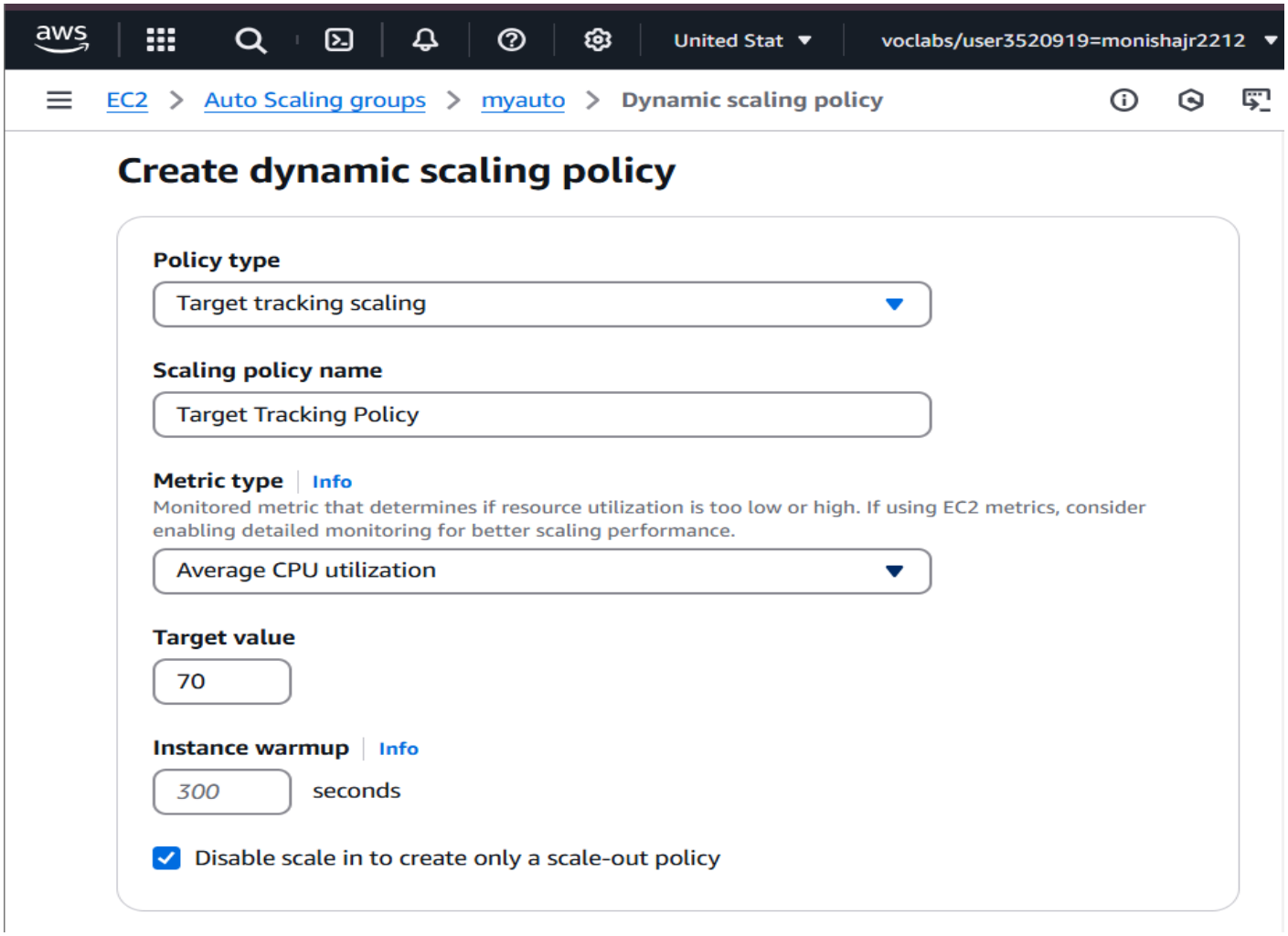


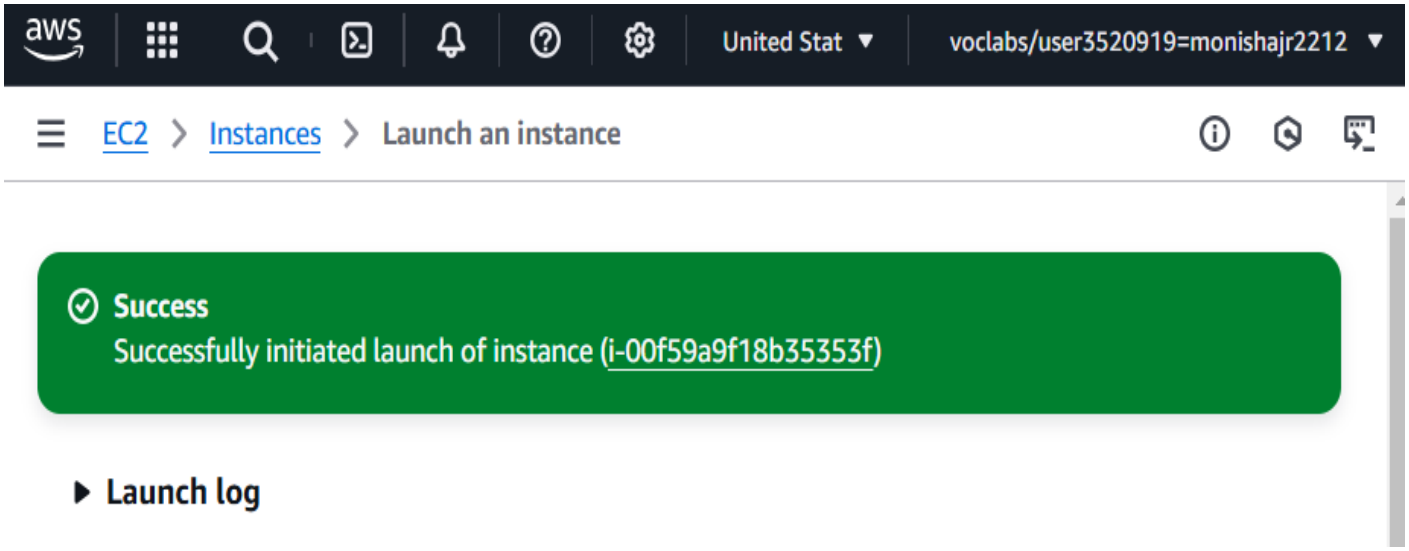


Step 3: Define Scaling Policies

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




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


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:ouCTuzvueMIwJ8S0jSEBmWkmjVROxHVSWn/+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 ~]$
```

```
[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                  889 kB/s |  34 kB     00:00
--------------------------------------------------------------------------------
Total                                                   494 kB/s |  34 kB     00:00
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**.