

```
stress --cpu 4  
yes > /dev/null &
```

Get started with Amazon EC2 Auto Scaling

[PDFRSS](#)

To get started with Amazon EC2 Auto Scaling, you can follow tutorials that introduce you to the service.

Topics

- [Tutorial: Create your first Auto Scaling group](#)
- [Tutorial: Set up a scaled and load-balanced application](#)

For additional tutorials that focus on specific tools for managing the lifecycle of instances in an Auto Scaling group, see the following topics:

- [Tutorial: Configure a lifecycle hook that invokes a Lambda function](#). This tutorial shows you how to use Amazon EventBridge to create rules that invoke Lambda functions based on events that happen to the instances in your Auto Scaling group.
- [Tutorial: Use data script and instance metadata to retrieve lifecycle state](#). This tutorial shows you how to use the Instance Metadata Service (IMDS) to invoke an action from within the instance itself.

Before you create an Auto Scaling group for use with your application, review your application thoroughly as it runs in the AWS Cloud. Consider the following:

- How many Availability Zones the Auto Scaling group should span.
- What existing resources can be used, such as security groups or Amazon Machine Images (AMIs).
- Whether you want to scale to increase or decrease capacity, or if you just want to ensure that a specific number of servers are always running. Keep in mind that Amazon EC2 Auto Scaling can do both simultaneously.

- What metrics have the most relevance to your application's performance.
- How long it takes to launch and provision a server.

The better you understand your application, the more effective you can make your Auto Scaling architecture.

<https://www.geeksforgeeks.org/create-and-configure-the-auto-scaling-group-in-ec2/>

Create and Configure the Auto Scaling Group in EC2

Last Updated : 17 Nov, 2023

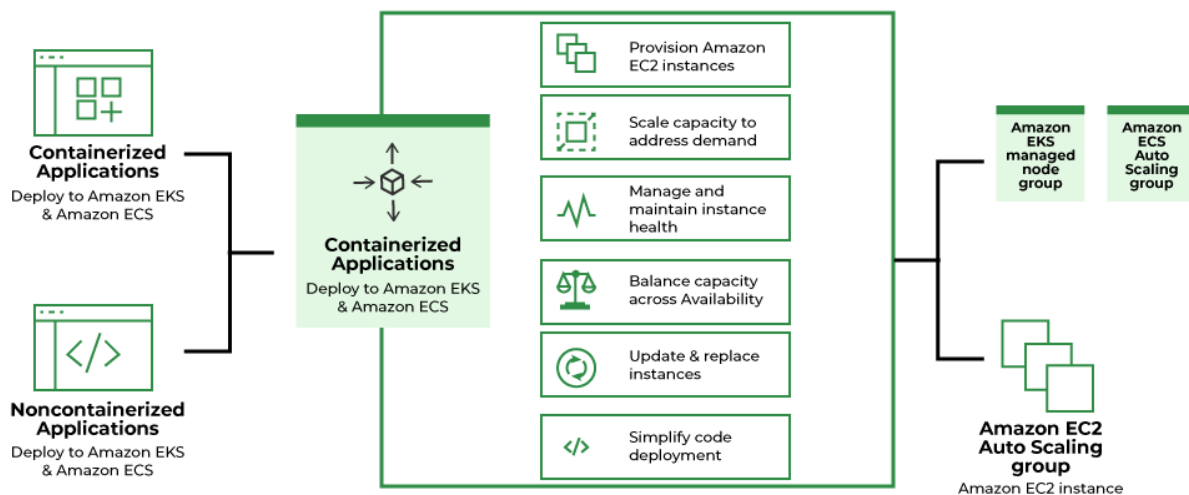


[Auto Scaling](#) is an Amazon Web Service it allows instances to scale when traffic or CPU load increases. Auto-scaling is a service that monitors all instances that are configured into the Auto Scaling group and ensures that loads are balanced in all instances. Depending on the load scaling group, increase the instance according to the configuration. When we created the auto-scaling group, we configured the Desired capacity, Minimum capacity, maximum capacity, and CPU utilization. If CPU utilization increases by 60% in all instances, one more instance is created, and if CPU utilization decreases by 30% in all instances, one instance is terminated. These are totally up to us; what is our requirement. If any Instance fails due to any reason, then the Scaling group maintains the Desired capacity and starts another instance.

The auto-scaling group follows Horizontal Scaling. This service is very important for us nowadays because we do not need to create new instances manually and do not require manual monitoring.

AWS Auto Scaling

AWS auto scaling is used to scale up and scale down the [EC2-instance](#) by depending up on the incoming traffic. You can scale up and scale down the applications in few minutes based up on the traffic which will decrease the latency of the application to the end-users. You can integrate the AWS Auto Scaling with multiple services provided by the AWS like Amazon traffic,, [Amazon DynamoDB](#), and [Amazon Aurora](#). You can also decrease the cost of an application because of dynamic scaling. When there is traffic , only maximum resources are used other wise it will use minimum resources.

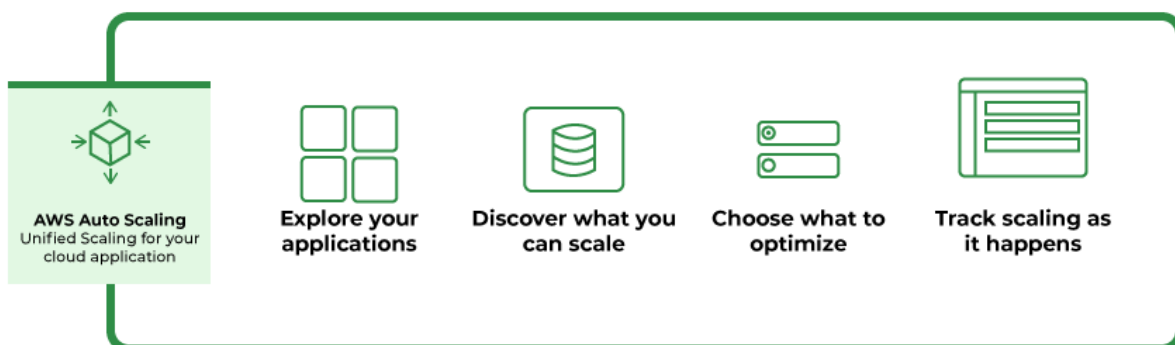


Benefits of Auto Scaling

1. **Dynamical scaling:** AWS auto-scaling service doesn't required any type of manual intervention it will automatically scale the application down and up by depending up on the incoming traffic.
2. **Pay For You Use:** Because of auto scaling the resource will be utilised in the optimised way where the demand is low the resource utilisation will be low and the demand will high the resource utilisation will increase so the AWS is going to charge you only for the amount of resources you really used.
3. **Automatic Performance Maintenance:** AWS autoscaling maintains the optimal application performance with considering the workloads it will ensures that the application is running to desired level which will decrease the latency and also the capacity will be increased by based on your application

How AWS Auto Scaling Works?

AWS autoscaling will scale the application based on the load of application. Instead of scaling manually AWS auto scaling will scale the application automatically when the incoming traffic is high it will scale up the application and when the traffic is low it will scale down the application.

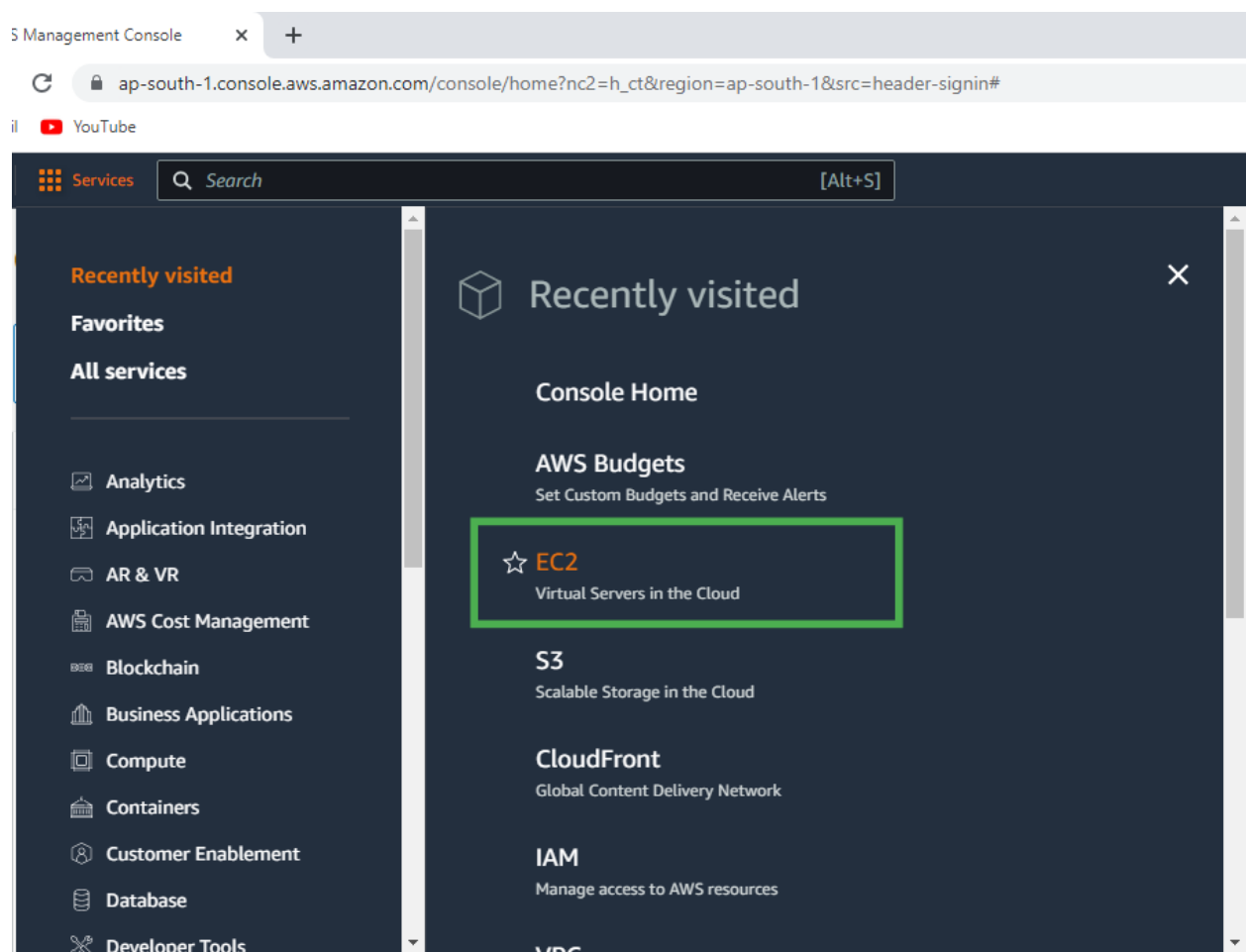


First you should choose which service or an application you want to scale then select the optimisation way like cost and performance and then keep track how the scaling is working.

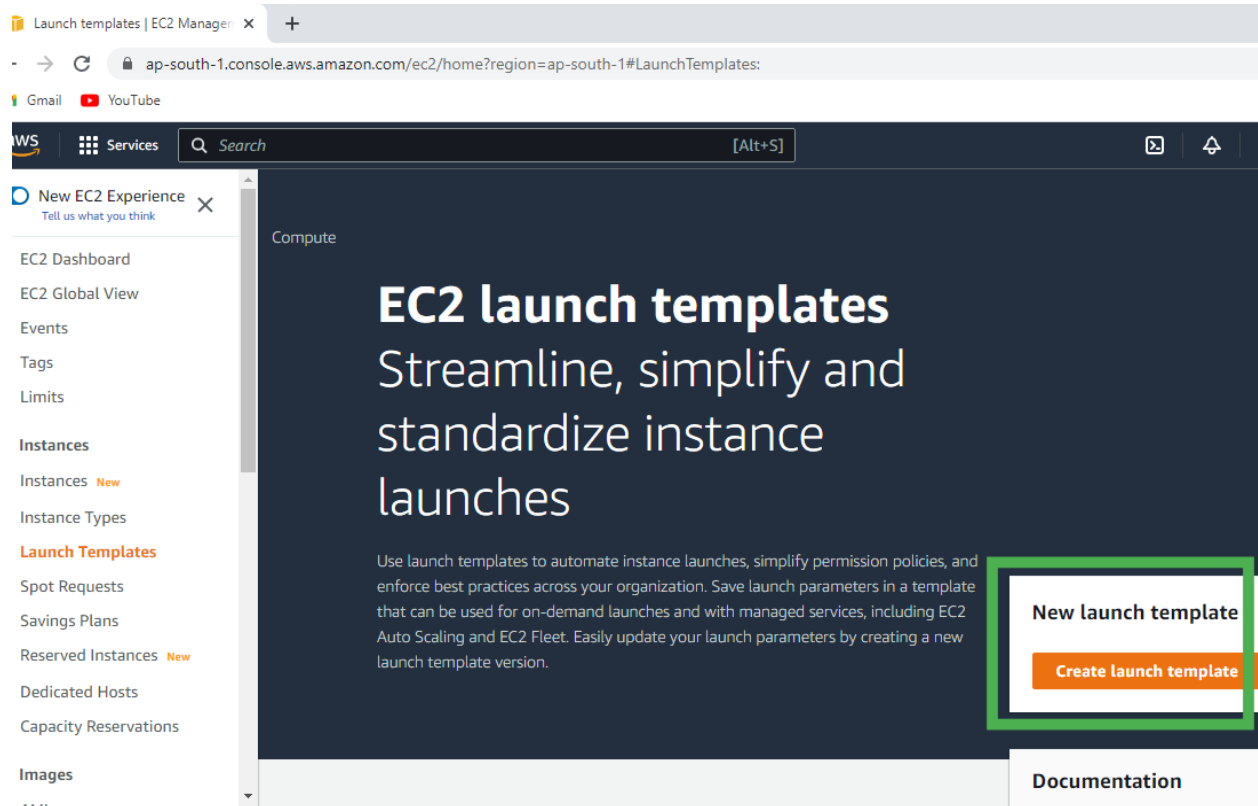
Steps To create Auto Scaling Launch Template

Step 1: Click on the All Services.

Step 2: Click on the **EC2(Elastic Cloud Computing)**.



Step 3: Scroll Down and click on the **Launch Templates** and click on the **Create launch template**



Step 4: Type the Template name.

Create launch template | EC2 Ma x +

← → ↻ ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateTemplate: Gmail YouTube

aws Services Search [Alt+S]

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*

my-template

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 [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

Step 5: Select the Amazon Machine Image.

Create launch template | EC2 Ma x +

← → ↻ ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateTemplate:

Gmail YouTube

aws Services Search [Alt+S]

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

▼ Application and OS Images (Amazon Machine Image) Info

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 Quick Start

Don't include in launch template

Amazon Linux
aws

macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

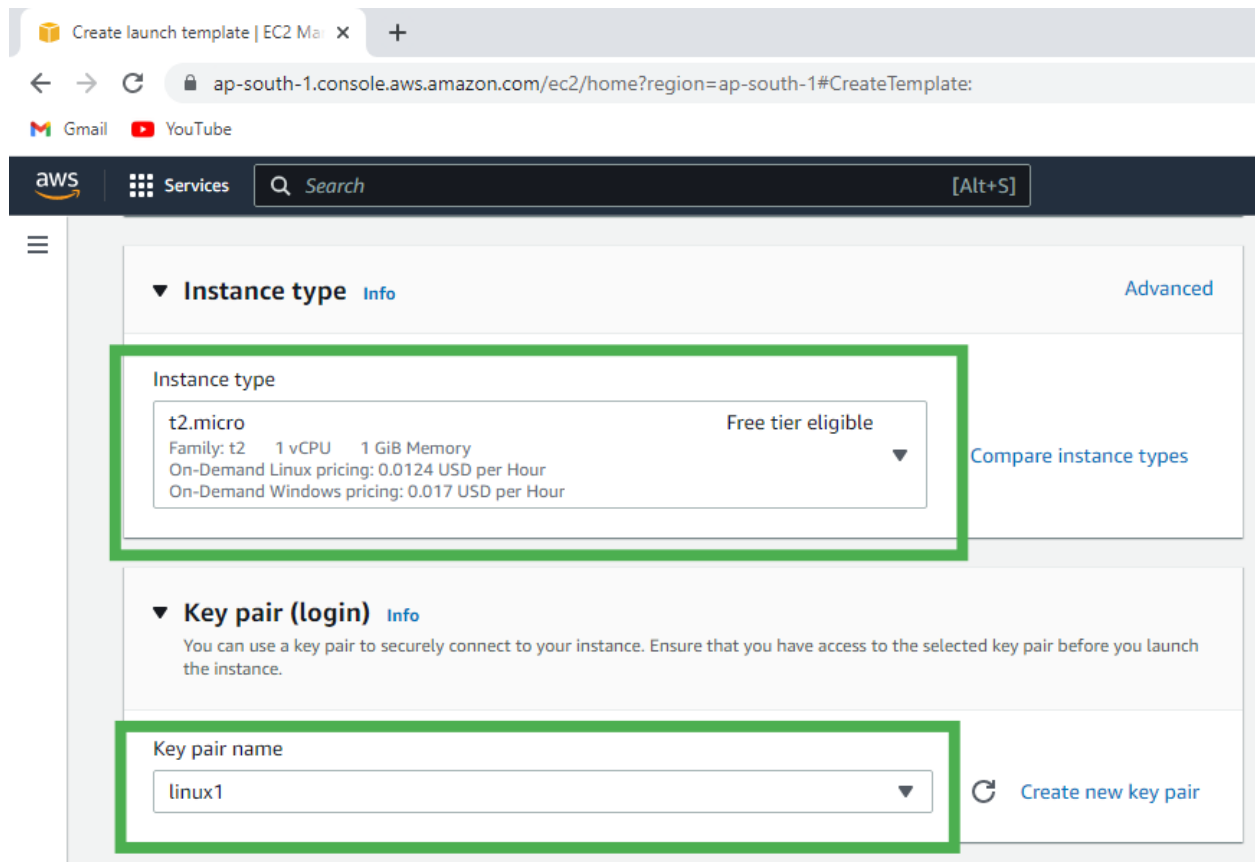
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

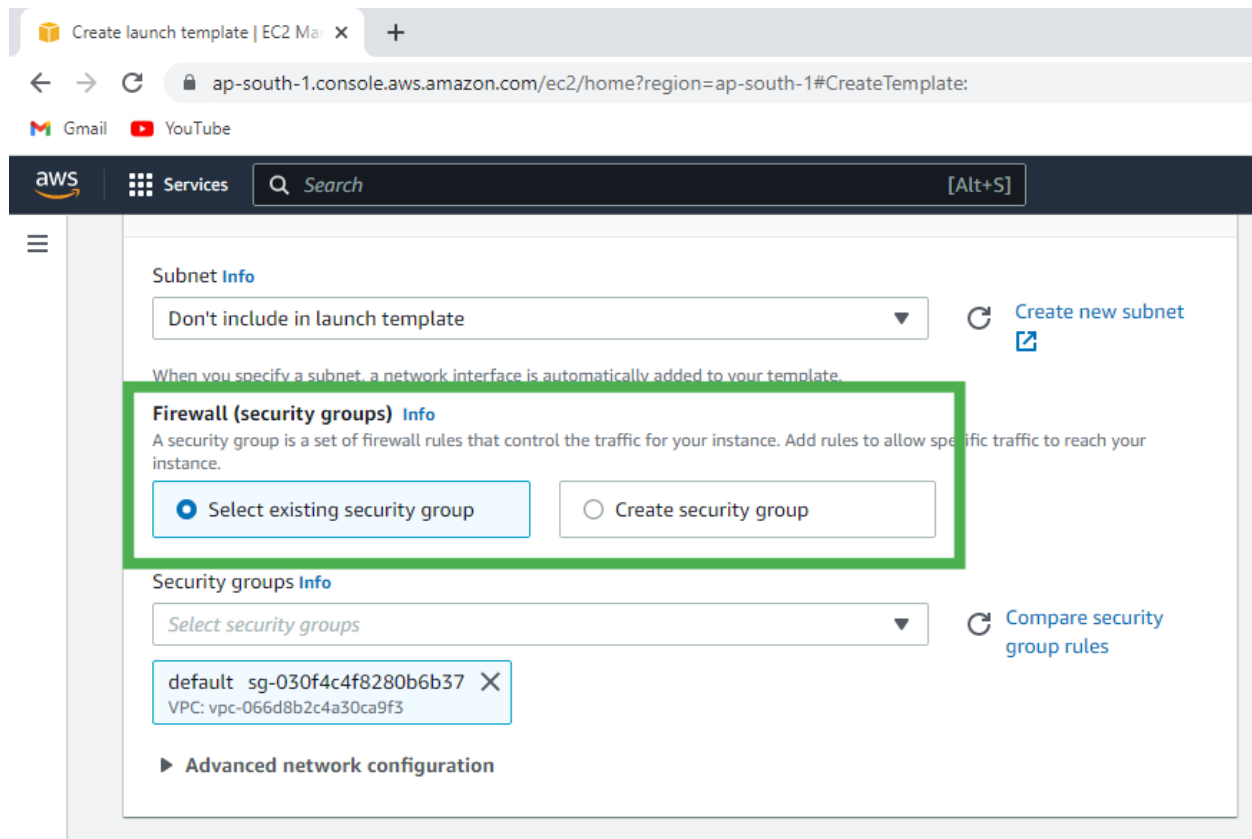
Amazon Linux 2 AMI (HVM) - Kernel 5.10 SSD Volume Type
ami-074dc0a6f6c764218 (64-bit (x86)) / ami-074e5caffd1685 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible ▼

Step 6: Select the Instance Type and Key pair.



Step 7: Select the **Security Group** or Create the new one.



Step 8: Click on the **Create Launch Template**.

Launch template | EC2 Ma x +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateTemplate:

YouTube

Services Search [Alt+S]

▼ Storage (volumes) Info

EBS Volumes

Hide details

▶ Volume 1 (AMI Root) (8 GiB, EBS, General purpose SSD (gp2))
AMI Volumes are not included in the template unless modified

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

Add new volume

▼ Resource tags Info

No resource tags are currently included in this template. Add a resource tag to include it in the launch template.

Add tag

50 remaining (Up to 50 tags maximum)

▼ Summary


Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI...[read n](#)
ami-074dc0a6f6c764218

Virtual server type (instance type)
t2.micro

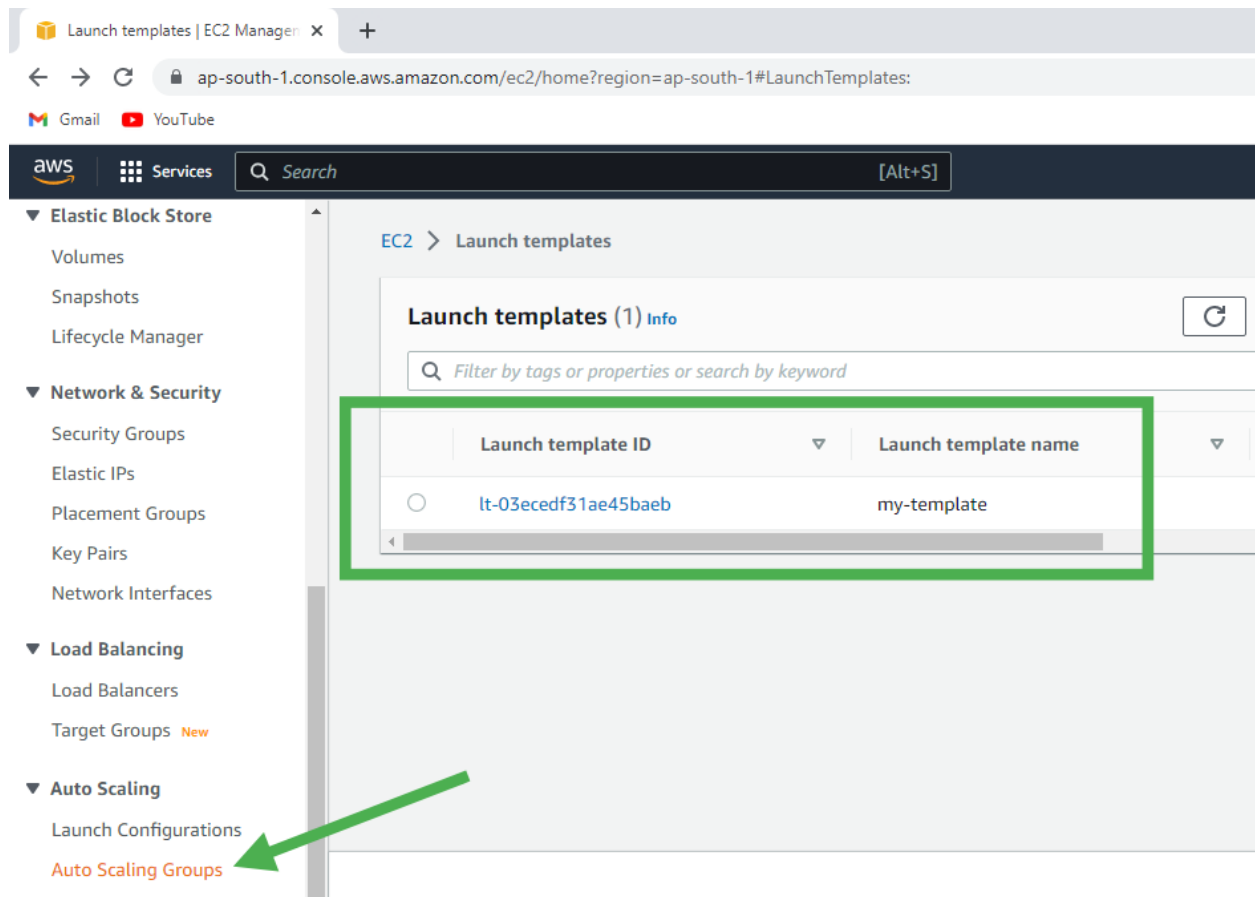
Firewall (security group)
default

Storage (volumes)
1 volume(s) - 8 GiB

Cancel Create launch

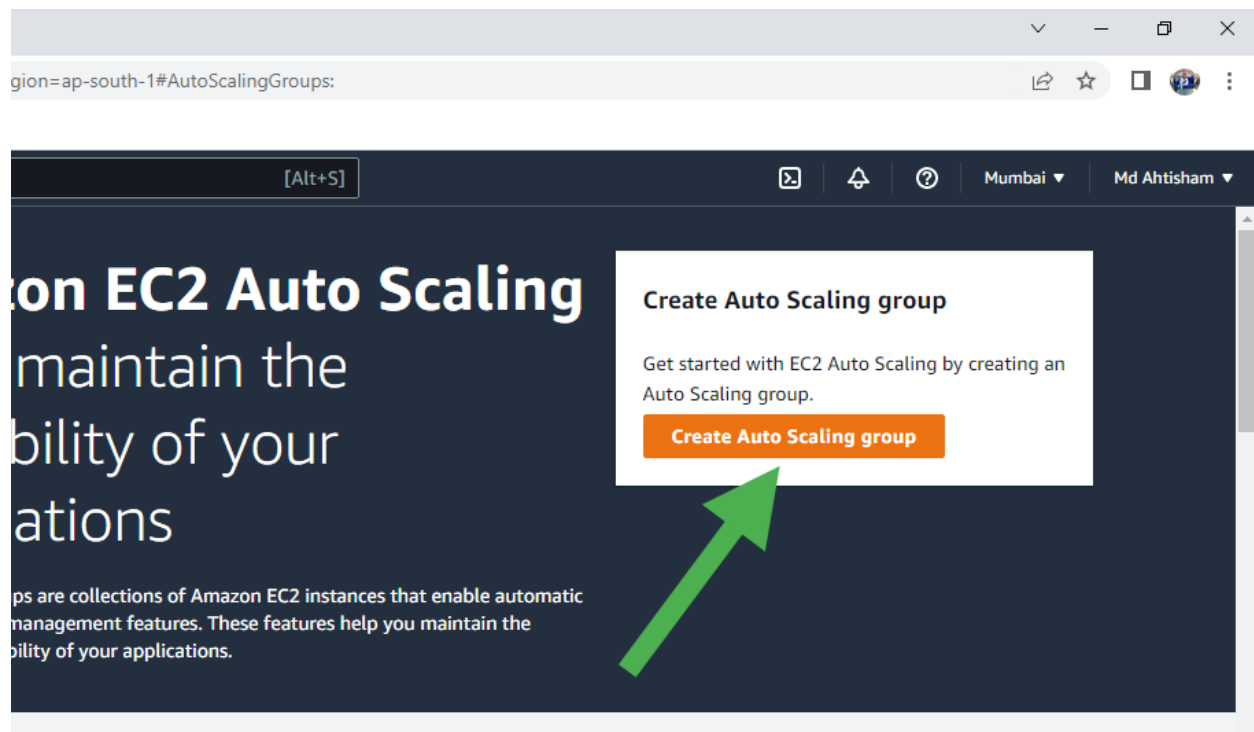


Step 9: Now you can see the template is **created**. Now, scroll down and click on the **Auto Scaling Groups**.



Create An Auto Scaling Group Using a Launch Template

Step 1: Click on the **Create Auto Scaling group**.



Step 2: Type the **Auto Scaling group name**.

Auto Scaling group | EC2 x +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateAutoScalingGroup:

YouTube

Services Search [Alt+S]

Step 1
Choose launch template or configuration

Step 2
Choose instance launch options

Step 3 (optional)
Configure advanced options

Step 4 (optional)
Configure group size and scaling policies

Step 5 (optional)
Add notifications

Step 6 (optional)
Add tags

Step 7

Choose launch template or configuration [Info](#)

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name

Auto Scaling group name
Enter a name to identify the group.

my-scaling

Must be unique to this account in the current Region and no more than 255 characters.

Launch template [Info](#) [Switch to launch configuration](#)

Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

Select a launch template ▼

Create a launch template [↗](#)

Step 3: Select your Template.

Step 5 (optional)
Add notifications

Step 6 (optional)
Add tags

Step 7
Review

Launch template [Info](#)

[Switch to launch configuration](#)

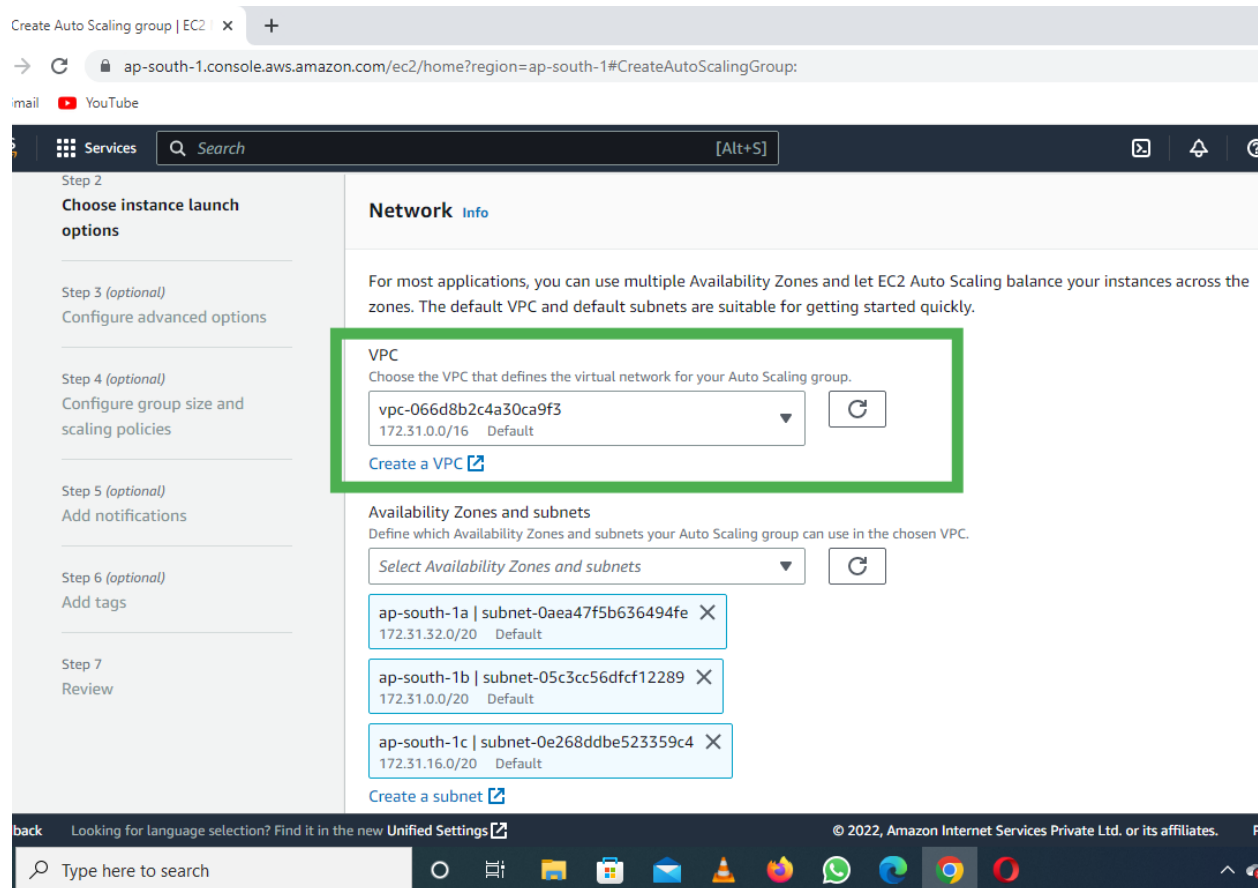
Launch template
Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups.

my-template ▼ [↻](#)

Version
Default (1) [↻](#)
[Create a launch template version](#)

Description -	Launch template my-template ↗ lt-03ecedf31ae45baeb	Instance type t2.micro
AMI ID ami-074dc0a6f6c764218	Security groups -	Request Spot Instances No
Key pair name linux1	Security group IDs sg-030f4c4f8280b6b37 ↗	

Step 4: Select the **VPC** or go with the default VPC and also select the **Availability zone**.



Step 5: Configure the **Group size** and **Scaling policies**.

Select as per your requirement:

- Desired: 4
- Minimum: 4
- Maximum: 8

ite Auto Scaling group | EC2 | x +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateAutoScalingGroup:

YouTube

Services Search [Alt+S]

Step 1
Choose launch template or configuration

Step 2
Choose instance launch options

Step 3 (optional)
Configure advanced options

Step 4 (optional)
Configure group size and scaling policies

Step 5 (optional)
Add notifications

Step 6 (optional)
Add tags

Step 7

Configure group size and scaling policies [Info](#)

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

Group size - optional [Info](#)

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity
4

Minimum capacity
4

Maximum capacity
8

Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Internet Services Private Ltd. or its affiliates.

Type here to search

Step 6: Select the Target tracking scaling policy.

Auto Scaling group | EC2 | x +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#CreateAutoScalingGroup:

YouTube

Services Search [Alt+S]

Step 7
Review

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
Target Tracking Policy

Metric type
Average CPU utilization

Target value
50

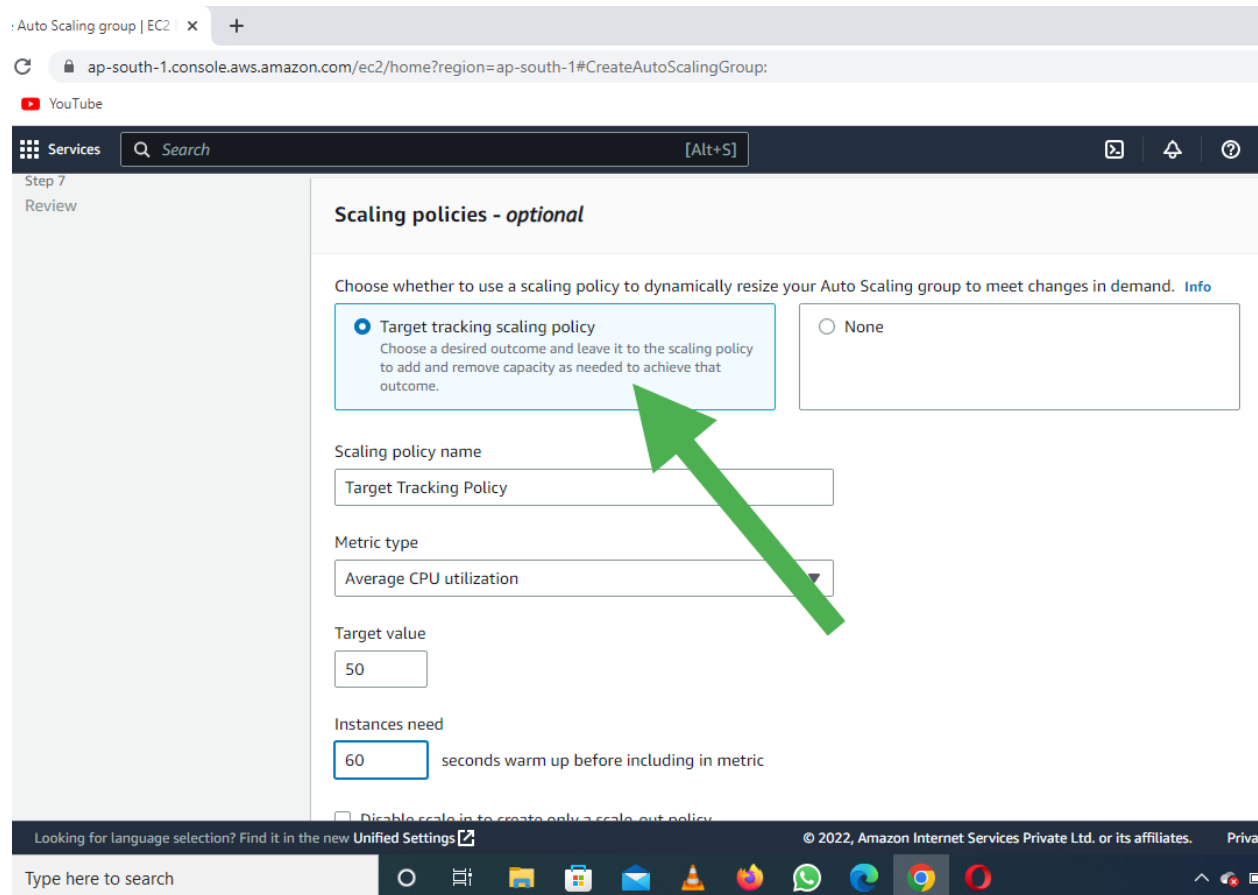
Instances need
60 seconds warm up before including in metric

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

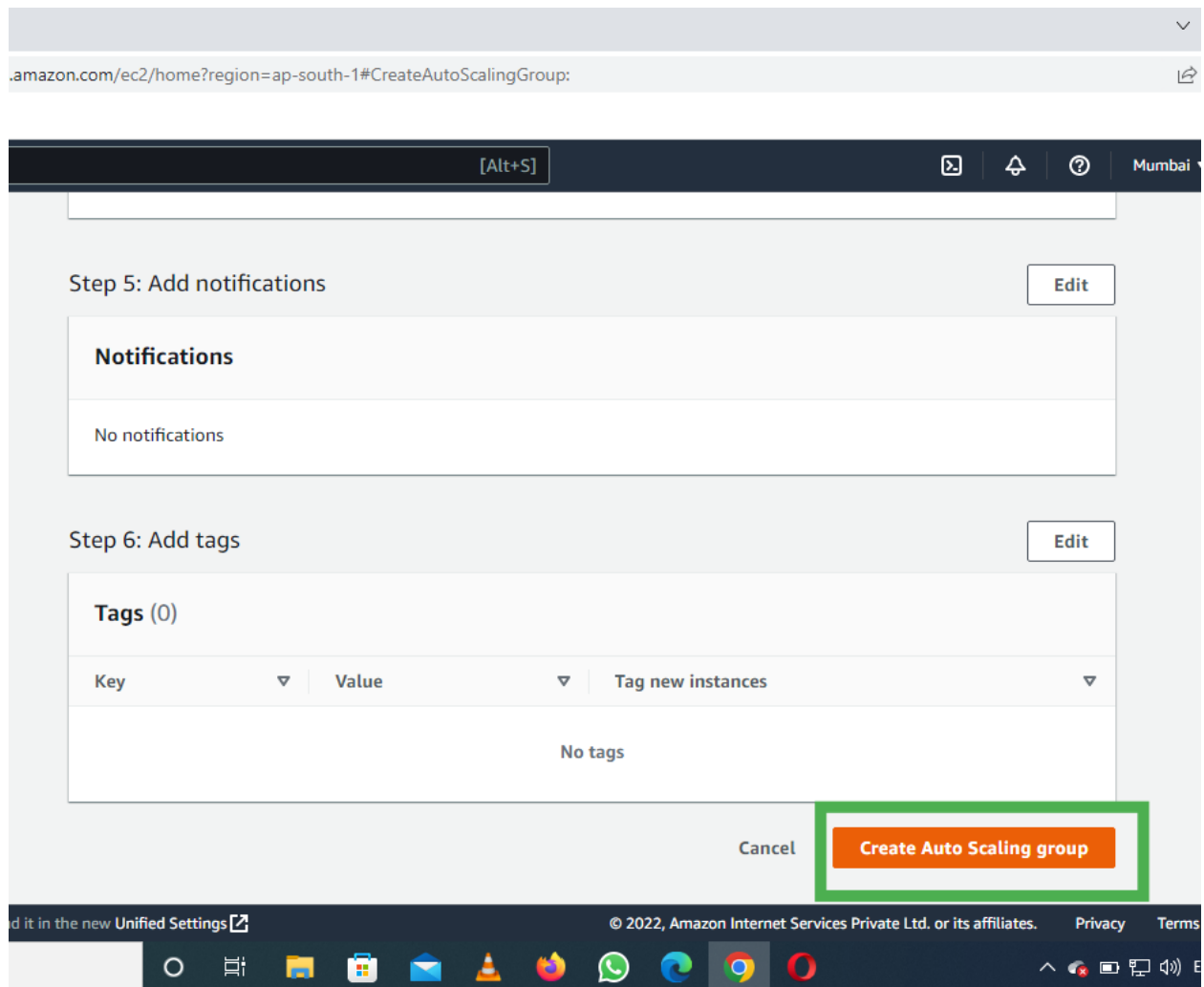
Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy

Type here to search



Step 7: Click on the **Create Auto Scaling Group**.



- Now you can see the **Auto Scaling is creating** and it is also creating the desired state of the EC2 Instance

Scaling groups | EC2 Manag x +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#AutoScalingGroups:

YouTube

Services Search [Alt+S]

my-scaling, 1 Scaling policy created successfully

EC2 > Auto Scaling groups

Auto Scaling groups (1) Info

Search your Auto Scaling groups

<input type="checkbox"/>	Name	Launch template/configuration	Instances	Status	Desired capacity
<input type="checkbox"/>	my-scaling	my-template Version Default	0	Updating capacity...	4

0 Auto Scaling groups selected

Looking for language selection? Find it in the new Unified Settings

Type here to search

© 2022, Amazon Internet Services Private Ltd. or its affiliates.

- We selected the **Desired state equal to 4** and you can see the **4 Instance is Running**

The screenshot displays the AWS Management Console interface for EC2 instances. At the top, the browser address bar shows the URL: `e.aws.amazon.com/ec2/home?region=ap-south-1#Instances:`. The console header includes a search bar with `[Alt+S]`, navigation icons, and the user's name `Md Ahtishar`. Below the header, the **Instances (4)** section is active, showing a list of four instances. A green box highlights the instance list table.

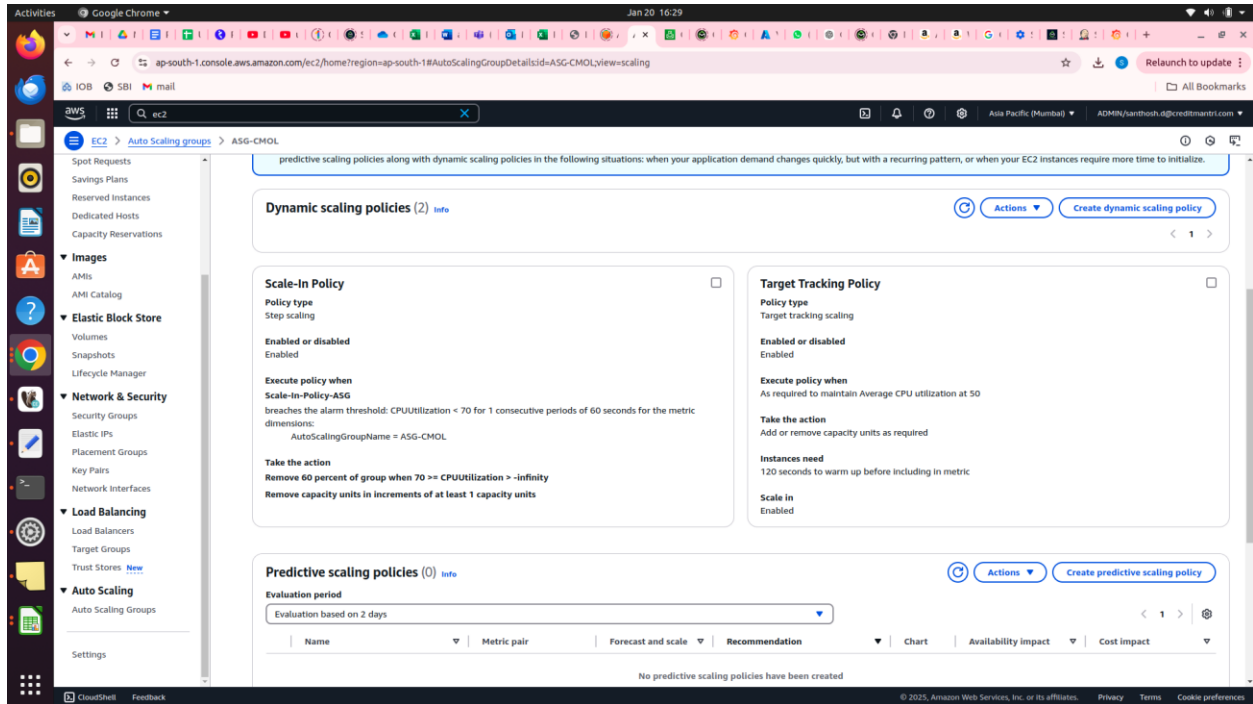
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability
<input type="checkbox"/>	-	i-0cec1bd4b24be62c5	Running	t2.micro	Initializing	No alarms	ap-south-1
<input type="checkbox"/>	-	i-0c2b1fd19e7935cde	Running	t2.micro	Initializing	No alarms	ap-south-1
<input type="checkbox"/>	-	i-0acfc6f7d4b3de8be	Running	t2.micro	Initializing	No alarms	ap-south-1
<input type="checkbox"/>	-	i-096b25590ff5d1293	Running	t2.micro	Initializing	No alarms	ap-south-1

Below the table, there is a section titled **Select an instance**. The footer of the console shows the copyright notice: `© 2022, Amazon Internet Services Private Ltd. or its affiliates.` and links for [Privacy](#), [Terms](#), and [Cookie preferences](#). The system clock indicates the time is 00:32 on 25-11-2022.

How AWS Target Tracking Policy Works

AWS Target Tracking Scaling Policies automatically adjust the capacity of an Auto Scaling Group (ASG) to maintain a specified target value for a chosen metric, such as **CPU Utilization** or **Request Count per Target**. The mechanism ensures that resources scale **out** or **in**

dynamically to keep the metric close to your defined target.



Key Components of Target Tracking Policy

1. **Target Metric:** The metric you want to monitor and maintain, such as:
 - a. **CPU Utilization**
 - b. **Network In/Out**
 - c. **Request Count per Target** (for ALB/ELB)
2. **Target Value:** The value you want the metric to achieve.
 - a. Example: Maintain an **average CPU Utilization of 50%** across instances.
3. **Scaling Actions:**

- a. **Scale Out:** Adds instances when the metric exceeds the target.
 - b. **Scale In:** Removes instances when the metric falls below the target.
- 4. **Cooldown Period:** The time AWS waits after a scaling action before evaluating further scaling actions.
- 5. **Disable Scale-In (Optional):**
 - a. If enabled, only scaling out actions will occur; scale-in actions will not happen.

How It Works (Step-by-Step)

1. **Monitor Metric:**
 - a. AWS CloudWatch continuously monitors the selected metric for your Auto Scaling Group.
2. **Evaluate Against Target:**
 - a. The policy compares the actual value of the metric to the **target value**.
3. **Trigger Scaling Actions:**
 - a. If the actual metric **exceeds the target value**, the policy triggers **scale-out** actions (adds instances).

- b. If the actual metric **falls below the target value**, the policy triggers **scale-in** actions (removes instances).

4. **Proportional Adjustment:**

- a. AWS adjusts the capacity of the ASG proportionally to bring the metric closer to the target.
- b. For example:
 - i. If CPU Utilization is 80% (target: 50%), AWS will add enough instances to bring the average utilization closer to 50%.
 - ii. If CPU Utilization is 20% (target: 50%), AWS will remove instances to bring it closer to 50%.

5. **Cooldown Period:**

- a. After a scaling action, AWS waits for the **cooldown period** to complete before reevaluating the metric to prevent unnecessary scaling actions during temporary fluctuations.

Example Configuration

Let's say you configure a Target Tracking Policy with the following settings:

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

- **Cooldown Period:** 300 seconds (5 minutes)

Scale-Out Scenario:

- Current average CPU utilization: **70%**
- AWS adds one or more instances to the ASG to reduce CPU utilization toward 50%.
- After scaling, AWS waits for 300 seconds (cooldown period) before re-evaluating the metric.

Scale-In Scenario:

- Current average CPU utilization: **30%**
- AWS removes one or more instances from the ASG to increase CPU utilization toward 50%.
- After scaling, AWS waits for 300 seconds (cooldown period) before re-evaluating the metric.

How Target Tracking Differs from Step Scaling

- **Target Tracking:**
 - Simple to configure.

- Automatically adjusts capacity to maintain the target metric.
- Proportional scaling actions based on deviation from the target.
- **Step Scaling:**
 - Requires manual configuration of thresholds and actions.
 - Executes predefined actions (e.g., add/remove X instances) when a threshold is crossed.

Best Practices for Target Tracking

1. Choose the Right Metric:

- a. Use a metric that accurately reflects your application's performance (e.g., CPU Utilization, Request Count).

2. Set Realistic Target Values:

- a. Avoid setting target values too high or too low, which could lead to excessive scaling actions.

3. Optimize Cooldown Period:

- a. Choose a cooldown period that allows enough time for scaling actions to stabilize before reevaluating.

4. Monitor ASG Activity:

- a. Review the **Scaling Activity** in the AWS Management Console to ensure the policy is working as expected.

Instance Scale-In Protection is a feature in AWS Auto Scaling that helps you prevent specific EC2 instances from being terminated during a scale-in operation. By enabling this protection, you ensure that critical instances remain running, even when the Auto Scaling group is reducing its size.

How to Enable Scale-In Protection and scale-out

You can enable instance scale-in protection either during the instance launch or for existing instances using the AWS Management Console, AWS CLI, or SDKs.

Using the AWS Management Console:

1. Navigate to EC2 Auto Scaling Groups:

- a. Open the [EC2 console](#).
- b. Go to **Auto Scaling Groups** in the left-hand navigation pane.

2. Select Your Auto Scaling Group:

- a. Click the name of the Auto Scaling group (ASG) containing the instance(s) you want to protect.

3. Manage Instance Scale-In Protection:

- a. Under the **Instance Management** tab, find the instance(s) you want to protect.
- b. Select the instance, then choose **Actions > Set Instance Protection > Protect from scale-in**.

4. Save Changes:

- a. Click **Save** to apply the scale-in protection.

Auto Scaling Group (ASG) Default Instance Warmup

Default Instance Warmup in AWS Auto Scaling ensures that new instances are fully initialized and ready to handle traffic before the scaling policy can trigger another scale-out event. This prevents over-scaling caused by metrics temporarily spiking during instance initialization.

Key Features of Default Instance Warmup:

1. **Cooldown Period:** Ensures that newly launched instances have time to stabilize before scaling policies act again.
2. **Metrics Stabilization:** Disables the scaling policies during the warmup period, so they do not act on metrics until the new instance is fully operational.
3. **Custom Warmup Period:** You can specify a custom duration (in seconds) for your instance warmup based on your application initialization time.

How to Configure Default Instance Warmup

Using AWS Management Console:

1. **Navigate to EC2 Auto Scaling Groups:**
 - a. Open the [EC2 console](#).
 - b. Select **Auto Scaling Groups** from the left-hand menu.

2. Choose Your Auto Scaling Group:

- a. Click on the name of the Auto Scaling group.

3. Edit Default Instance Warmup:

- a. Go to the **Automatic scaling** tab.
- b. Click **Edit** under **Scaling policies**.
- c. Locate the **Default instance warmup** section and specify the warmup time in seconds (e.g., 300 seconds for 5 minutes).

4. Save Changes:

- a. Click **Save** to apply the configuration.

Create AMI:-

```
1.aws ec2 create-image --instance-id i-013c4cb47635f943c --name "MyServerAMI" --description "Backup of my server" --no-reboot --region ap-south-1
```

create-launch-template-version:-

```
2.aws ec2 create-launch-template-version --launch-template-id lt-00ee9ff021f0d0dbb --source-version 1 --launch-template-data "{\"ImageId\":\"ami-0fd5f9dc78cb14f43\"}" --query
```

```
'LaunchTemplateVersion.VersionNumber' --output text --  
region ap-south-1
```

```
3. aws autoscaling update-auto-scaling-group \  
    --auto-scaling-group-name my-auto-scaling-group \  
    --launch-template "LaunchTemplateId=lt-  
0abcd1234efgh5678,Version=$NEW_VERSION"
```

autoscaling start-instance-refresh:-

```
4.aws autoscaling start-instance-refresh --auto-scaling-  
group-name ASG-CMOL --region ap-south-1
```

Explanation of the Script:

This script automates the process of creating a new Amazon Machine Image (AMI), updating the launch template, updating the Auto Scaling Group (ASG) with the new AMI, and initiating an instance refresh in the Auto Scaling Group. Below is a detailed breakdown of each section of the script:

1. Variable Initialization:

bash

Copy code

Variables

INSTANCE_ID="i-013c4cb47635f943c"

REGION="ap-south-1"

AMI_NAME="MyServerAMI"

AMI_DESCRIPTION="Backup of my server"

LAUNCH_TEMPLATE_ID="lt-00ee9ff021f0d0dbb"

SOURCE_VERSION="1"

ASG_NAME="ASG-CMOL"

- **INSTANCE_ID:** The ID of the EC2 instance you want to create an AMI from.
- **REGION:** The AWS region where your instance, launch template, and Auto Scaling Group are located.
- **AMI_NAME:** The name to assign to the new AMI.
- **AMI_DESCRIPTION:** A brief description of the new AMI.
- **LAUNCH_TEMPLATE_ID:** The ID of the launch template that you will update with the new AMI.
- **SOURCE_VERSION:** The source version of the launch template that is being updated.

- **ASG_NAME**: The Auto Scaling Group's name that you want to update.

2. Create the AMI:

bash

Copy code

Create the AMI

```
AMI_ID=$(aws ec2 create-image \
  --instance-id "$INSTANCE_ID" \
  --name "$AMI_NAME" \
  --description "$AMI_DESCRIPTION" \
  --no-reboot \
  --region "$REGION" \
  --query 'ImageId' \
  --output text)
```

- **aws ec2 create-image**: This AWS CLI command creates an AMI from the specified EC2 instance.
 - **--no-reboot**: Ensures the instance is not rebooted during the AMI creation process.
 - **--query 'ImageId'**: Extracts only the AMI ID from the command output.
 - **--output text**: Ensures that the output is in plain text format (AMI ID).

- The **AMI_ID** is captured into a variable to be used later.

3. Check if AMI Creation is Successful:

bash

Copy code

```
# Check if the AMI creation was successful
if [ $? -ne 0 ]; then
    echo "Failed to create AMI."
    exit 1
else
    echo "AMI created successfully. AMI ID: $AMI_ID"
fi
```

- **\$?**: This is a special variable that holds the exit status of the last executed command.
 - If the AMI creation was successful, the exit status will be 0.
 - If there was an error, the exit status will be non-zero (indicating failure).
- Based on the exit status, a success or failure message is displayed, and the script exits if there was an error.

4. Update the Launch Template:

bash

Copy code

Update the launch template with the new AMI

```
LT_VERSION=$(aws ec2 create-launch-template-version \
  --launch-template-id "$LAUNCH_TEMPLATE_ID" \
  --source-version "$SOURCE_VERSION" \
  --launch-template-data "{\"ImageId\":\"$AMI_ID\"}" \
  --query 'LaunchTemplateVersion.VersionNumber' \
  --output text \
  --region "$REGION")
```

- **aws ec2 create-launch-template-version:** This command creates a new version of the launch template with the new AMI ID.
 - **--launch-template-id:** Specifies the ID of the launch template to be updated.
 - **--source-version:** Specifies the version of the launch template to base the new version on.
 - **--launch-template-data:** Specifies the new AMI ID to be used in the launch template.
- The **LT_VERSION** variable captures the new launch template version number.

5. Check if Launch Template Update is Successful:

bash

Copy code

```
# Check if the launch template update was successful
if [ $? -ne 0 ]; then
    echo "Failed to update the launch template."
    exit 1
else
    echo "Launch template updated successfully. New
Version Number: $LT_VERSION"
fi
```

- Similar to the AMI creation check, the script checks the exit status to determine whether the launch template update was successful or not.

6. Update the Auto Scaling Group (ASG):

bash

Copy code

```
# Update the Auto Scaling Group with the new launch
template version
aws autoscaling update-auto-scaling-group \
    --auto-scaling-group-name "$ASG_NAME" \
    --launch-template
"LaunchTemplateId=$LAUNCH_TEMPLATE_ID,Version=$L
```

```
T_VERSION" \  
--region "$REGION"
```

- **aws autoscaling update-auto-scaling-group**: This command updates the Auto Scaling Group with the new launch template version, using the new AMI.
 - **--auto-scaling-group-name**: The name of the Auto Scaling Group to be updated.
 - **--launch-template**: Specifies the launch template ID and the version to be used by the Auto Scaling Group.

7. Check if ASG Update is Successful:

```
bash
```

```
Copy code
```

```
# Check if the ASG update was successful  
if [ $? -ne 0 ]; then  
    echo "Failed to update the Auto Scaling Group."  
    exit 1  
else  
    echo "Auto Scaling Group updated successfully with  
Launch Template Version: $LT_VERSION"  
fi
```

- Again, checks if the ASG update was successful using \$? and prints a success or failure message accordingly.

8. Start Instance Refresh in Auto Scaling Group:

bash

Copy code

```
# Start an Instance Refresh
```

```
aws autoscaling start-instance-refresh \  
  --auto-scaling-group-name "$ASG_NAME" \  
  --region "$REGION"
```

- **aws autoscaling start-instance-refresh:** This command triggers a rolling instance refresh in the Auto Scaling Group.
 - This causes the ASG to replace instances with the updated launch template that contains the new AMI.
 - It ensures that new instances are launched with the updated configuration, and old instances are terminated gradually.

9. Check if Instance Refresh Was Started Successfully:

bash

Copy code

```
# Check if the Instance Refresh was started successfully
if [ $? -ne 0 ]; then
    echo "Failed to start the Instance Refresh."
    exit 1
else
    echo "Instance Refresh started successfully for Auto
Scaling Group: $ASG_NAME"
fi
```

- The script checks if the start-instance-refresh command was successful using \$? and outputs the appropriate message.

Summary of Workflow:

1. **Create AMI:** A backup image of the EC2 instance is created.
2. **Update Launch Template:** The launch template is updated to reference the new AMI.
3. **Update Auto Scaling Group:** The Auto Scaling Group is updated to use the new launch template version.
4. **Start Instance Refresh:** A rolling update is triggered, and old instances are replaced with new ones using the updated AMI.

This ensures that your Auto Scaling Group is using the latest AMI, and all instances are refreshed with the new configuration automatically.

Before configuring Auto Scaling in AWS, certain prerequisites and steps should be followed for both **Scale-In** and **Scale-Out** actions. Here's an overview of the necessary setup:

Prerequisites for Auto Scaling Group (ASG) Configuration:

- 1. Launch Configuration/Template:**

- a. Launch Template or Launch Configuration**

defines the configuration for the EC2 instances that will be launched as part of the Auto Scaling group.

- i. This includes instance type, AMI (Amazon Machine Image), security group, key pair, and other configurations.

- b. Action:** Make sure your launch template has the correct configurations, including the AMI, instance type, and other required parameters.

- 2. Auto Scaling Group (ASG):**

- a. The **Auto Scaling Group** defines the desired number of EC2 instances to run and their associated settings (like VPC, Subnet, and Scaling Policies).
- b. **Action:** Create an Auto Scaling Group and attach the launch template or configuration to it. Set up desired capacity, minimum, and maximum instance count.

3. **CloudWatch Alarms:**

- a. **CloudWatch Alarms** monitor your EC2 instances and trigger scale-in or scale-out actions based on CPU utilization, network traffic, or any other custom metrics.
- b. **Action:** Create appropriate CloudWatch Alarms for scaling actions, such as CPU usage exceeding 80% (for scale-out) or falling below 20% (for scale-in).

4. **Scaling Policies:**

- a. Scaling Policies define the conditions and actions that will trigger scaling, either scaling out (adding instances) or scaling in (removing instances).
 - i. **Scale-Out:** Adding EC2 instances when your application load increases.
 - ii. **Scale-In:** Removing EC2 instances when the load decreases.

- b. **Action:** Define the scaling policies based on CloudWatch Alarms.

5. IAM Role:

- a. Make sure that your EC2 instances and Auto Scaling Group have appropriate **IAM roles** to allow AWS Auto Scaling to launch and terminate instances.
- b. **Action:** Ensure that the Auto Scaling group has the necessary permissions to interact with EC2 and CloudWatch services.

6. Elastic Load Balancer (ELB) (Optional but recommended):

- a. An **Elastic Load Balancer (ELB)** can distribute incoming traffic to instances in the Auto Scaling group. This is recommended to ensure high availability.
- b. **Action:** Attach the Load Balancer to your Auto Scaling Group, so new instances launched by the ASG can automatically register with the load balancer.

7. Health Checks:

- a. AWS Auto Scaling uses **Health Checks** to determine if instances are running as expected.
- b. **Action:** Set up **EC2 health checks** and optionally **ELB health checks** to ensure that only healthy instances are part of the Auto Scaling group.

Example: Auto Scaling Policy Setup

1. Create CloudWatch Alarms:

- a. **Scale-Out** Alarm (trigger when CPU usage is above 80%):

bash

Copy code

```
aws cloudwatch put-metric-alarm \  
  --alarm-name "ScaleOutAlarm" \  
  --metric-name "CPUUtilization" \  
  --namespace "AWS/EC2" \  
  --statistic "Average" \  
  --period 300 \  
  --threshold 80 \  
  --comparison-operator "GreaterThanThreshold" \  
  --dimension  
"Name=AutoScalingGroupName,Value=your-auto-scaling-  
group-name" \  
  --evaluation-periods 2 \  
  --alarm-actions "arn:aws:autoscaling:region:account-  
id:scalingPolicy:policy-id"
```

- b. **Scale-In** Alarm (trigger when CPU usage is below 20%):

bash

Copy code

```
aws cloudwatch put-metric-alarm \  
  --alarm-name "ScaleInAlarm" \  
  --metric-name "CPUUtilization" \  
  --namespace "AWS/EC2" \  
  --statistic "Average" \  
  --period 300 \  
  --threshold 20 \  
  --comparison-operator "LessThanThreshold" \  
  --dimension  
"Name=AutoScalingGroupName,Value=your-auto-scaling-  
group-name" \  
  --evaluation-periods 2 \  
  --alarm-actions "arn:aws:autoscaling:region:account-  
id:scalingPolicy:policy-id"
```

2. Create Scaling Policies:

- a. **Scale-Out Policy** (add one more instance when triggered):

bash

Copy code

```
aws autoscaling put-scaling-policy \  
  --auto-scaling-group-name "your-auto-scaling-group-  
name" \  

```

```
--policy-name "ScaleOutPolicy" \  
--scaling-adjustment 1 \  
--adjustment-type "ChangeInCapacity" \  
--cooldown 300
```

b. **Scale-In Policy** (remove one instance when triggered):

bash

Copy code

```
aws autoscaling put-scaling-policy \  
  --auto-scaling-group-name "your-auto-scaling-group-  
name" \  
  --policy-name "ScaleInPolicy" \  
  --scaling-adjustment -1 \  
  --adjustment-type "ChangeInCapacity" \  
  --cooldown 300
```

3. Attach Scaling Policies to Alarms:

a. Attach the **Scale-Out Alarm** to the **Scale-Out Policy** and the **Scale-In Alarm** to the **Scale-In Policy** to trigger these actions when the alarms are met.

Final Steps:

1. **Test** your Auto Scaling setup:

- a. Simulate load on your application and observe if the Auto Scaling Group scales in and scales out as expected.

2. **Monitor** Auto Scaling:

- a. Use CloudWatch dashboards and AWS Auto Scaling metrics to monitor the scaling behavior.

By following these steps, you can set up Auto Scaling in AWS with Scale-In and Scale-Out actions triggered based on CloudWatch alarms and scaling policies.

prerequisite for auto scale :-

step-1: create launch template

Launch template name and version description:-

Launch template name : CMOL-API-Template (lt-00489838eee06dd55)

Launch template contents:-

Amazon Machine Image (AMI): ami-059562647cb0289a7

Instance type :-

Instance type: t3a.xlarge

Key pair (login) :-

Key pair name : cmprod

Network settings :-

Select existing security group: Bastion-Access-Only-LIVE,Bastion-80-443,CM-Monitoring

Advanced details :-

IAM instance profile: snoopy-cmol-backup

step-2: Create Auto Scaling Groups

ASG-CMOL Capacity overview:-

Name: ASG-CMOL

Group size :

Desired capacity: 2

Min desired capacity:2

Max desired capacity: 6

Launch template:-

attach launch template:

