

# AWS- Auto Scaling Practical

Siddesh Mandhare || 10/09/24

**Name:** - Autoscaling ensures your application stays available by adjusting resources as needed.

## Step 1: - Create launch template configuration.

1. Choose Amazon Linux as the AMI.
2. Select the t2.micro instance type.
3. Choose a key pair.
4. Configure the security group. (efs\_demo)
5. Assign a IAM role with access of S3FullAccess.
6. In advanced settings, add a bootstrap script. (Copy S3 file)
7. Create the launch template.

The screenshot displays the AWS Management Console interface for a launch template. The browser address bar shows the URL: <https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchTemplateDetails:launchTemplateId=lt-0fb5e9d7f70b8e15c>. The console header includes the AWS logo, a search bar, and the user's name 'siddesh\_mandhare'.

The left-hand navigation pane lists various services under categories like 'Instances', 'Images', 'Elastic Block Store', and 'Network & Security'. The main content area is titled 'My\_Launch (lt-0fb5e9d7f70b8e15c)' and contains two sections:

- Launch template details:** A table showing the launch template ID (lt-0fb5e9d7f70b8e15c), name (My\_Launch), default version (1), and owner (arn:aws:iam::985539783646:root).
- Launch template version details:** A section for version 1 (Default) showing its description, date created (2024-09-12T15:26:48.000Z), and creator (arn:aws:iam::985539783646:root).

Below these sections, there are tabs for 'Instance details', 'Storage', 'Resource tags', 'Network interfaces', and 'Advanced details'. The 'Instance details' tab is active, showing a table with the AMI ID (ami-0e53db6fd757e38c7), instance type (t2.micro), availability zone, and key pair name (Siddesh).

The bottom of the image shows a Windows taskbar with the date and time set to 20:58 on 12/09/2024.

## Step 2: - Create Autoscaling Group

### 1) Provide group name and select template which is created

The screenshot shows the AWS Management Console interface for creating an Auto Scaling group. The page is titled 'Create Auto Scaling group' and is at Step 1: 'Choose launch template'. The left sidebar shows the navigation menu with 'Auto Scaling groups' selected. The main content area has a 'Name' field with the value 'Auto\_scaling' and a 'Launch template' dropdown menu with the value 'My\_Launch'. A blue information box states: '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.' Below the dropdown, there is a 'Version' dropdown set to 'Default (1)' and a 'Description' field with the value 'My\_Launch'. The 'Instance type' is set to 't2.micro'. The bottom of the page shows the AWS CloudShell toolbar and the Windows taskbar.

Step 1  
Choose launch template

Step 2  
Choose instance launch options

Step 3 - optional  
Configure advanced options

Step 4 - optional  
Configure group size and scaling

Step 5 - optional  
Add notifications

Step 6 - optional  
Add tags

Step 7  
Review

Choose launch template

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group.

Name

Auto Scaling group name

Enter a name to identify the group.

Auto\_scaling

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

Launch template

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.

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\_Launch

Create a launch template

Version

Default (1)

Create a launch template version

Description

My\_Launch

Instance type

t2.micro

AMI ID

Security groups

Request Spot Instances

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

24°C Clear

Search

ENG IN

21:02 12/09/2024

### 2) Select Availability Zone which you want to create instance

The screenshot shows the AWS Management Console interface for creating an Auto Scaling group, Step 2: 'Choose instance launch options'. The left sidebar shows the navigation menu with 'Auto Scaling groups' selected. The main content area has a 'Launch template' dropdown set to 'My\_Launch' and a 'Version' dropdown set to 'Default'. The 'Instance type' is set to 't2.micro'. The 'Network' section is expanded, showing a 'VPC' dropdown set to 'vpc-0e1949b8cd950dd9' and a 'Subnet' dropdown set to 'ap-south-1a | subnet-056a9faf4162ea00f'. A red box highlights the 'Availability Zones and subnets' section, which includes a 'Select Availability Zones and subnets' dropdown and a list of subnets: 'ap-south-1a | subnet-056a9faf4162ea00f' and 'ap-south-1b | subnet-069f4bde34103fab9'. The bottom of the page shows the AWS CloudShell toolbar and the Windows taskbar.

Step 1  
Choose launch template

Step 2  
Choose instance launch options

Step 3 - optional  
Configure advanced options

Step 4 - optional  
Configure group size and scaling

Step 5 - optional  
Add notifications

Step 6 - optional  
Add tags

Step 7  
Review

Choose instance launch options

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

Instance type requirements

You can keep the same instance attributes or instance type from your launch template, or you can choose to override the launch template by specifying different instance attributes or manually adding instance types.

Override launch template

Launch template

My\_Launch

Version

Default

Description

-

Instance type

t2.micro

Network

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0e1949b8cd950dd9

Create a VPC

Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets

ap-south-1a | subnet-056a9faf4162ea00f

ap-south-1b | subnet-069f4bde34103fab9

Create a subnet

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

24°C Clear

Search

ENG IN

21:02 12/09/2024

### 3) Configured group size and scaling as per requirement

**Configure group size and scaling - optional**

Define your group's desired capacity and scaling limits. You can optionally add automatic scaling to adjust the size of your group.

**Group size**

Set the initial size of the Auto Scaling group. After creating the group, you can change its size to meet demand, either manually or by using automatic scaling.

**Desired capacity type**

Choose the unit of measurement for the desired capacity value. vCPUs and MemoryGB are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances)

**Desired capacity**

Specify your group size.

3

**Scaling**

You can resize your Auto Scaling group manually or automatically to meet changes in demand.

**Scaling limits**

Set limits on how much your desired capacity can be increased or decreased.

**Min desired capacity**

2

**Max desired capacity**

5

**Automatic scaling - optional**

Choose whether to use a target tracking policy.

☒ No scaling policies

☐ Target tracking scaling policy

### 4) Autoscaling group have been created

**Autoscaling**

**Details** | Activity | Automatic scaling | Instance management | Monitoring | Instance refresh

**Group details**

Auto Scaling group name Autoscaling	Desired capacity 3	Desired capacity type Units (number of instances)	Amazon Resource Name (ARN) arn:aws:autoscaling:ap-south-1:985539783646:autoScalingGroup:autoScalingGroup/pa0140fba-19cd-4f5d-9d19-8a1d723f3ee:autoScalingGroupName/Autoscaling
Date created Thu Sep 12 2024 21:40:27 GMT+0530 (India Standard Time)	Minimum capacity 2	Status Updating capacity	
	Maximum capacity 5		

**Launch template**

Launch template lt-0fb5e9d770b0e15c My_Launch	AMI ID ami-0e53db6fd757e38c7	Instance type t2.micro	Owner arn:aws:iam::985539783646:root
Version Default	Security groups -	Security group IDs sg-0c1687896dc38393f	Create time Thu Sep 12 2024 20:56:48 GMT+0530 (India Standard Time)
Description -	Storage (volumes) -	Key pair name Siddeh	Request Spot Instances No

**Network**

### 5) Autoscaling group have been created 3 instances as we mentioned in desire "3" count

**Instances (4)**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs
	i-0eb81b0d4ac0560a	Terminated	t2.micro	-	View alarms	ap-south-1a	-	-	-	-
	i-0091a226733e660c8	Running	t2.micro	Initializing	View alarms	ap-south-1a	ec2-13-235-45-149.ap-...	13.235.45.149	-	-
	i-0f90c48bc799f73f	Running	t2.micro	2/2 checks pass	View alarms	ap-south-1b	ec2-45-2-122-242.ap-s...	65.2.122.242	-	-
	i-0131e4e4eaa20a4b7	Running	t2.micro	2/2 checks pass	View alarms	ap-south-1b	ec2-5-110-56-127.ap-s...	5.110.56.127	-	-

## Manual scaling

**Step 1: - If I manually change the desired value to 2 from 3, the Autoscaling group will remove 1 instance**

### Group size

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum scaling limits.

**Desired capacity type**  
Choose the unit of measurement for the desired capacity value. vCPUs and Memory(GiB) are only supported for mixed instances groups configured with a set of instance attributes.

Units (number of instances) ▼

**Desired capacity**  
Specify your group size.

2

**Scaling limits**  
Set limits on how much your desired capacity can be increased or decreased.

**Min desired capacity**  
2  
Equal or less than desired capacity

**Max desired capacity**  
5  
Equal or greater than desired capacity

Cancel Update

**Step 2: - We can see one instance is going to terminate**

Instances (4) Info											
Find Instance by attribute or tag (case-sensitive)											
All states ▼											
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs
<input type="checkbox"/>		i-0eb81cdda0560e	Terminated	t2.micro	-	View alarms +	ap-south-1a	-	-	-	-
<input type="checkbox"/>		i-0091a226733e660c8	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1a	ec2-13-235-45-149.ap-...	13.235.45.149	-	-
<input type="checkbox"/>		i-0fcb0c4abc799f73f	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-65-2-122-242.ap-s...	65.2.122.242	-	-
<input type="checkbox"/>		i-0131e4e4aa20a4b7	Shutting-d...	t2.micro	-	View alarms +	ap-south-1b	ec2-3-110-56-127.ap-s...	3.110.56.127	-	-

## Automatic scaling

We required alarm to set create “Dynamic scaling policies”

### Step 1: - Create alarm for Autoscaling group (Increase)

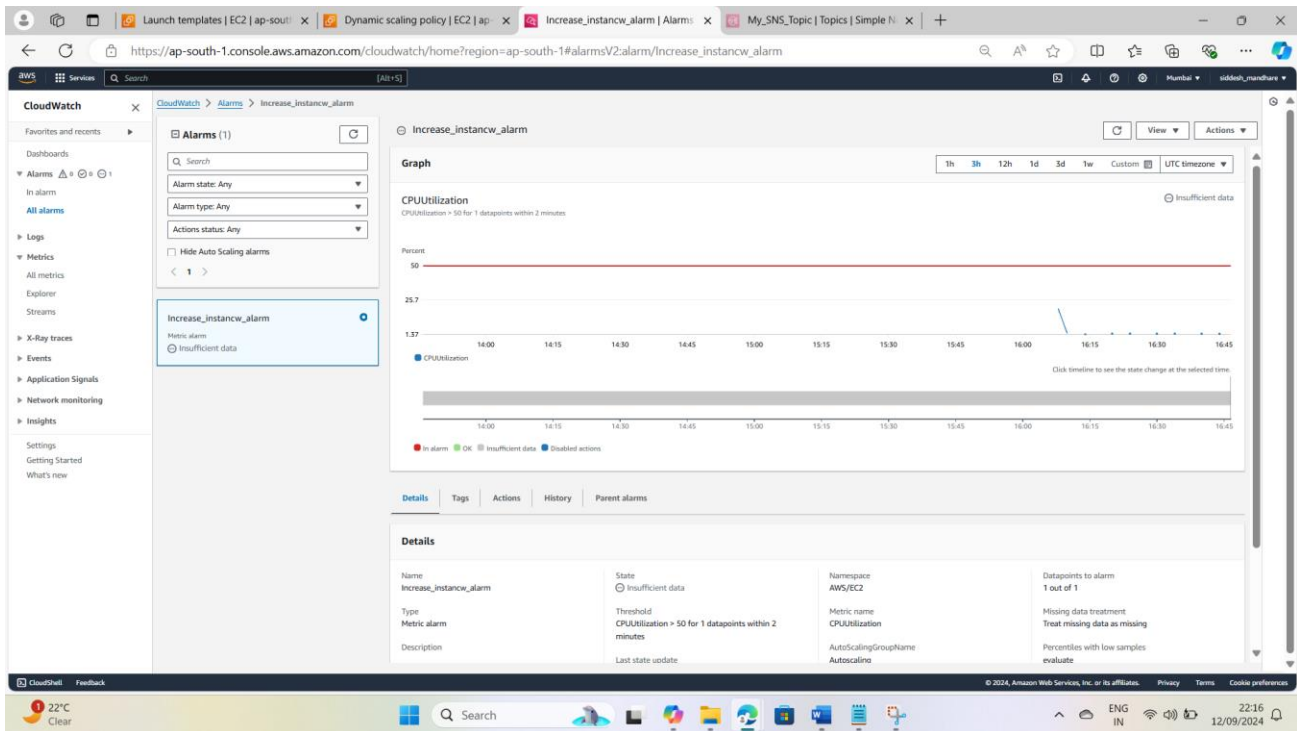
1) Select metrics (CPU Utilization) and time period “1 min” and threshold value (utilization>50)

The screenshot shows the 'Specify metric and conditions' step in the AWS CloudWatch console. The left sidebar lists steps: Step 1 (Specify metric and conditions), Step 2 (Configure actions), Step 3 (Add name and description), and Step 4 (Preview and create). The main area is divided into 'Metric' and 'Conditions' sections. The 'Metric' section includes a graph showing CPU Utilization over time, with a red dashed line at 50%. A red box highlights the configuration fields: Namespace (AWS/EC2), Metric name (CPUUtilization), AutoScalingGroupName (Autoscaling), Statistic (Average), and Period (2 minutes). The 'Conditions' section shows 'Threshold type' set to 'Static' and 'Whenever CPUUtilization is...' set to 'Greater' than the threshold value '50'. A red box highlights the '50' threshold value field.

2) Select in which state you want to trigger “In alarm” and add SNS topic

The screenshot shows the 'Configure actions' step in the AWS CloudWatch console. The left sidebar lists steps: Step 1 (Specify metric and conditions), Step 2 (Configure actions), Step 3 (Add name and description), and Step 4 (Preview and create). The main area is the 'Notification' section. Under 'Alarm state trigger', the 'In alarm' option is selected. Below, 'Send a notification to the following SNS topic' is selected, and 'My\_SNS\_Topic' is chosen from the dropdown. The 'Email (endpoints)' section shows 'sidmandhare3@gmail.com' with a link to 'View in SNS Console'. An 'Add notification' button is at the bottom.

### 3) Alarm is created successfully



### 4) Create Dynamic Policy with alarm in take action added count as "1". It means if utilization is goes above threshold i.e. 50% it should add one more instance.

EC2 > Auto Scaling groups > Autoscaling

## Create dynamic scaling policy

Policy type  
Simple scaling

Scaling policy name  
Increase\_instance

CloudWatch alarm  
Choose an alarm that can scale capacity whenever:  
Increase\_instancw\_alarm

Create a CloudWatch alarm [Create a CloudWatch alarm](#)

breaches the alarm threshold: CPUUtilization > 50 for 1 consecutive periods of 120 seconds for the metric dimensions:

AutoScalingGroupName = Autoscaling

Take the action  
Add 1 capacity units

And then wait  
300 seconds before allowing another scaling activity

Cancel Create

## 5) Created Dynamic policy as “Increase\_instance”

The screenshot displays the AWS Management Console's 'Autoscaling' page for an 'Auto Scaling group details | EC2'. The 'Automatic scaling' tab is selected, showing a list of 'Dynamic scaling policies (1)'. The policy 'Increase\_instance' is highlighted, showing its configuration:

- Policy type:** Simple scaling
- Enabled or disabled:** Enabled
- Execute policy when:** Increase\_instancw\_alarm breaches the alarm threshold: CPUUtilization > 50 for 1 consecutive periods of 120 seconds for the metric dimensions: AutoScalingGroupName = Autoscaling
- Take the action:** Add 1 capacity units
- And then wait:** 300 seconds before allowing another scaling activity

The console interface includes a left-hand navigation menu with categories like Instances, Images, Elastic Block Store, Network & Security, and Load Balancing. The top navigation bar shows the user's location (Mumbai) and session details (siddeh\_mandhare).

**Step 2: - Create Alarm for one more dynamic policy we need to create one more alarm as “Decrease\_instance” Now condition is (if CPU utilization is below 20% it should trigger the alarm)**

### 1) Create alarm

The screenshot shows the 'Create alarm' wizard in the AWS Management Console, specifically the 'Specify metric and conditions' step. The 'Metric' section displays a graph for 'CPUUtilization' with a threshold line at 19.7%. The configuration details on the right are as follows:

- Namespace:** AWS/EC2
- Metric name:** CPUUtilization
- AutoScalingGroupName:** Autoscaling
- Statistic:** Average
- Period:** 2 minutes

The left-hand navigation menu shows the steps of the wizard: Step 1 (Specify metric and conditions), Step 2 (Configure actions), Step 3 (Add name and description), and Step 4 (Preview and create).

## Conditions

### Threshold type

☒ Static

Use a value as a threshold

☐ Anomaly detection

Use a band as a threshold

### Whenever CPUUtilization is...

Define the alarm condition.

☐ Greater

> threshold

☐ Greater/Equal

>= threshold

☐ Lower/Equal

<= threshold

☒ Lower

< threshold

### than...

Define the threshold value.

20

Must be a number

► Additional configuration

## 2) Alarm is created

The screenshot shows the AWS CloudWatch console interface. The top navigation bar includes the AWS logo, Services, Search, and user information. The main content area is titled 'Step 1: Specify metric and conditions' and includes a 'Metric' section with a graph and a 'Conditions' section. The graph shows CPUUtilization over time, with a red line at 20% and a blue line below it. The conditions section shows 'Threshold type: Static', 'Whenever CPUUtilization is: Lower (<)', and 'than...: 20'. The 'Additional configuration' section is also visible.

Below the main content area, a green banner indicates 'Successfully created alarm Decrease\_instance\_alarm.' with a 'View alarm' link. The bottom section shows the 'Alarms (2)' list, which includes the following details:

Name	State	Last state update (UTC)	Conditions	Actions
Decrease_instance_alarm	Insufficient data	2024-09-12 17:05:49	CPUUtilization < 20 for 1 datapoints within 2 minutes	Actions enabled
Increase_instance_alarm	OK	2024-09-12 16:47:11	CPUUtilization > 50 for 1 datapoints within 2 minutes	Actions enabled



**Step 3: - Create policy as “decrease\_instance” in take action select “Remove” and count as “1” It means if utilization is below its threshold i.e. 20% it should remove one instance**

EC2 > Auto Scaling groups > Autoscaling

### Create dynamic scaling policy

Policy type  
Simple scaling

Scaling policy name  
Decrease\_instance

CloudWatch alarm  
Choose an alarm that can scale capacity whenever:  
Decrease\_instance\_alarm [Create a CloudWatch alarm](#)  
breaches the alarm threshold: CPUUtilization < 20 for 1 consecutive periods of 120 seconds for the metric dimensions:  
AutoScalingGroupName = Autoscaling

Take the action  
Remove 1 capacity units

And then wait  
300 seconds before allowing another scaling activity

Cancel Create

❖ We have added two policies successfully and both are enabled.

Dynamic scaling policy created or edited successfully.

EC2 > Auto Scaling groups > Autoscaling

### Autoscaling

Details Activity Automatic scaling Instance management Monitoring Instance refresh

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 (2) Info

Decrease_instance	Increase_instance
<p>Policy type Simple scaling</p> <p>Enabled or disabled Enabled</p> <p>Execute policy when Decrease_instance_alarm breaches the alarm threshold: CPUUtilization &lt; 20 for 1 consecutive periods of 120 seconds for the metric dimensions: AutoScalingGroupName = Autoscaling</p> <p>Take the action Remove 1 capacity units</p>	<p>Policy type Simple scaling</p> <p>Enabled or disabled Enabled</p> <p>Execute policy when Increase_instance_alarm breaches the alarm threshold: CPUUtilization &gt; 50 for 1 consecutive periods of 120 seconds for the metric dimensions: AutoScalingGroupName = Autoscaling</p> <p>Take the action Add 1 capacity units</p>

## Step 4: -Output

We could see one alarm has been triggered “decrease\_instance\_alarm”. As checked average CPU utilization of instance is below 20% only.

CloudWatch

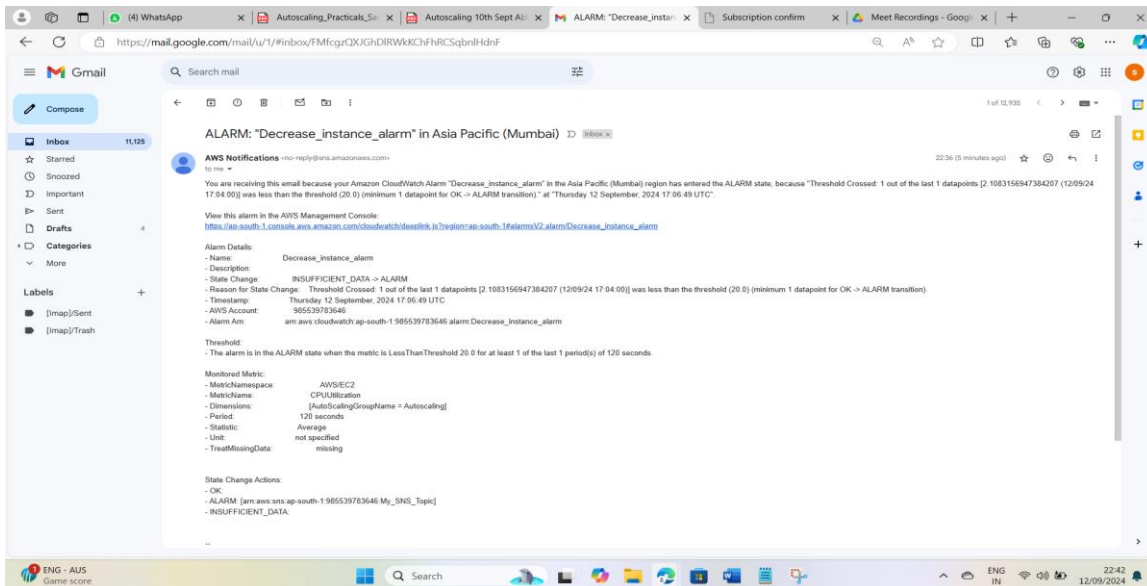
Successfully created alarm Decrease\_instance\_alarm. View alarm

CloudWatch > Alarms

Alarms (2)

Name	State	Last state update (UTC)	Conditions	Actions
Decrease_instance_alarm	In alarm	2024-09-12 17:06:49	CPUUtilization < 20 for 1 datapoints within 2 minutes	Actions enabled
Increase_instance_alarm	OK	2024-09-12 16:47:11	CPUUtilization > 50 for 1 datapoints within 2 minutes	Actions enabled

## ❖ We got notification mail



## Summary

- **Automatic Resource Management:** AWS Auto Scaling dynamically adjusts the number of EC2 instances or other resources in your application based on current demand. This ensures that you always have the right amount of resources to handle the load.
- **Improved Availability:** By automatically scaling up during high demand and scaling down during low demand, AWS Auto Scaling helps maintain the availability and performance of your application. This minimizes the risk of downtime and ensures a smooth user experience.
- **Cost Efficiency:** AWS Auto Scaling optimizes costs by automatically adjusting resources to match the current demand. This means you only pay for the resources you actually need, avoiding over-provisioning and reducing unnecessary expenses.
- **Customizable Policies:** You can define scaling policies based on specific metrics such as CPU utilization, memory usage, or custom CloudWatch metrics. This allows you to tailor the scaling behavior to the unique needs of your application.
- **Integration:** AWS Auto Scaling integrates seamlessly with other AWS services like EC2, ECS, and RDS. This makes it easy to manage and scale a wide range of resources within your AWS environment.
- **Health Monitoring:** AWS Auto Scaling continuously monitors the health of your instances. If an instance becomes unhealthy, it is automatically replaced with a new one, ensuring that your application remains reliable and resilient.
- **Scalability:** AWS Auto Scaling supports both horizontal scaling (adding more instances) and vertical scaling (increasing the size of instances). This flexibility allows you to efficiently manage resources as your application grows.
- **Predictive Scaling:** AWS Auto Scaling can use machine learning to predict future traffic patterns and scale your resources proactively. This helps ensure that your application is prepared for upcoming demand spikes.
- **Easy Setup and Management:** Setting up AWS Auto Scaling is straightforward, and the service provides a user-friendly interface for managing scaling policies and monitoring resource usage.