

Source: C# Corner (www.c-sharpcorner.com)[PRINT](#)

Article

Implementing Auto Scaling In AWS



By **Amit Tyagi** on Updated date Nov 21 2019

In this article we will be covering the below topics for the implementation of auto scaling in AWS. Here we will be covering the below topics:

- Introduction to Amazon EC2 Auto Scaling.
- Creating AMI for existing EC2 instance.
- Creating Classic Load Balancer.
- Need for auto Scaling.
- Setting up auto scaling group
 - Creating launch configuration.
 - Creating auto scaling group.
 - Choosing scaling policies.
- Deleting EC2 instances to check auto scaling.

INTRODUCTION TO AMAZON EC2 AUTO SCALING

Auto Scaling is a web service designed to launch or terminate Amazon EC2 instances automatically based on user-defined policies, schedules, and health checks.

With Auto Scaling we can make sure that we have the desired number of EC2 instances always running. It can also automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance.

Adding auto scaling to your existing application is one of the most beneficial way to make use of AWS cloud .It has many benefits like,

- It ensures that your application has the right capacity to handle the traffic.
- It can dynamically increase or decrease the capacity as per the need as thus it can reduce the overall cost.
- We can configure auto scaling to be used in multiple Availability zones to ensure the high availability even if one AZ is down.

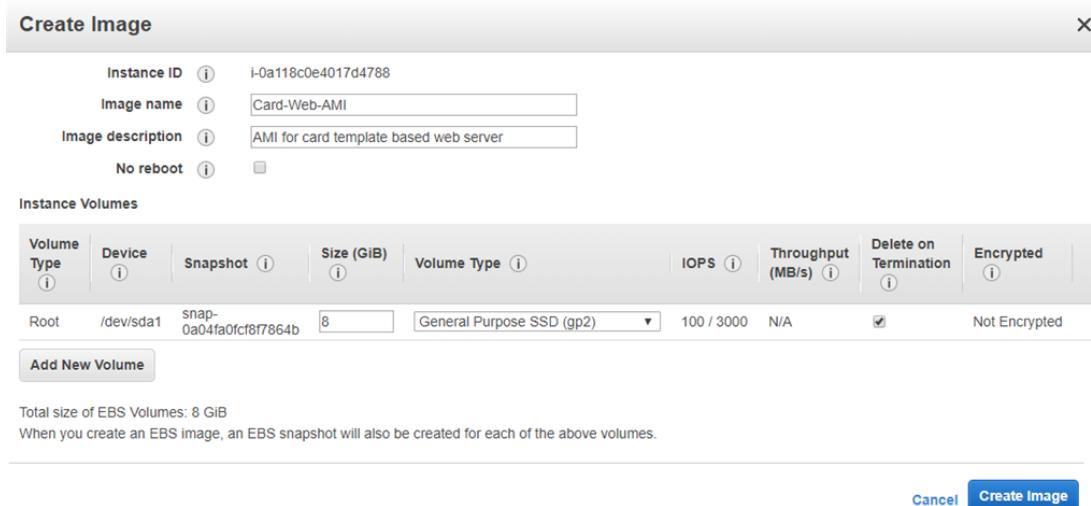
CREATING AMI FOR EXISTING EC2 INSTANCE

Now as we have gathered some idea around Auto Scaling, it's time to actually see its implementation. For this we are first going to create the AMI which we can use for auto scaling.

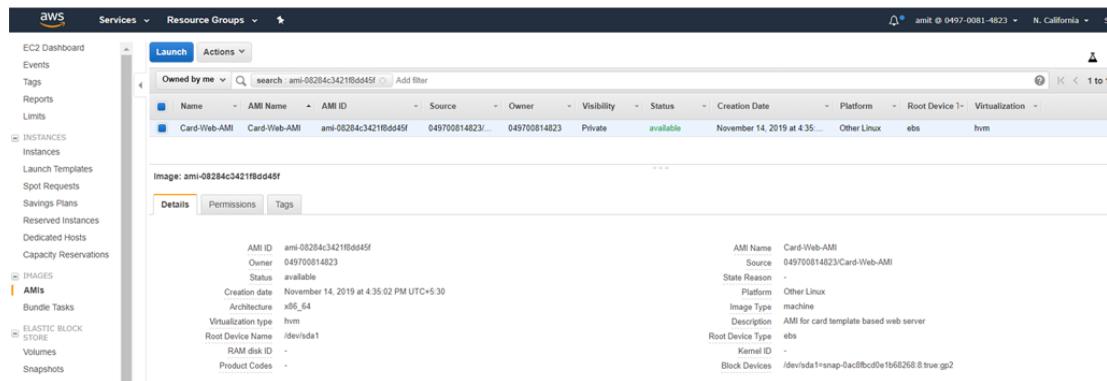
In one of the previous article [Clustering Web Servers Using ELB](#) we have seen how we can create the AMI from an existing EC2 instance.

So similarly, here also we have created the AMI for the existing instance **card-web01**, and the process has been covered in the previous articles.

Below are some of the screenshots.



Here we can see that our AMI has been created successfully.



CREATING CLASSIC LOAD BALANCER

Now we have our AMI ready which we named as Card-Web-AMI and has one EC2 instance card-web01. So next we have to create the load balancer to balance the load. In this article we are going to create the Classic Load Balancer. The *Classic Load Balancer* operates on both the request and connection levels. However, it doesn't support features like host and path-based routing.

This is the first load balancer that AWS introduced in 2009 so it is missing some features. The Application Load Balancer was introduced to address this. A Classic Load Balancer is recommended only for EC2 Classic instances.

Let's get started to create the classic load balancer.

The screenshot shows the 'Select load balancer type' section. It compares three types:

- Application Load Balancer:** Handles HTTP and HTTPS traffic. Description: Choose an Application Load Balancer when you need a flexible feature set for your web applications with HTTP and HTTPS traffic. Operating at the request level, Application Load Balancers provide advanced routing and visibility features targeted at application architectures, including microservices and containers.
- Network Load Balancer:** Handles TCP, TLS, and UDP traffic. Description: Choose a Network Load Balancer when you need ultra-high performance, TLS offloading at scale, centralized certificate deployment, support for UDP and static IP addresses for your application. Operating at the connection level, Network Load Balancers are capable of handling millions of requests per second securely while maintaining ultra-low latencies.
- Classic Load Balancer:** Previous generation for HTTP, HTTPS, and TCP. Description: Choose a Classic Load Balancer when you have an existing application running in the EC2-Classic network.

Here we have selected the classic Load balancer and has given it the name as card-web-prod-lb.

The screenshot shows the 'Step 1: Define Load Balancer' wizard. Under 'Basic Configuration', the 'Load Balancer name' is set to 'card-web-prod-lb'. The 'Create LB Inside' dropdown is set to 'My Default VPC (172.31.0.0/16)'. The 'Create an internal load balancer' checkbox is checked. The 'Enabled advanced VPC configuration' checkbox is checked. The 'Listener Configuration' table shows a single entry for port 80 on both the Load Balancer Port and Instance Port.

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	80

We have also given the front end and back end port as 80 as our HTTP request comes on port 80 and the service running in EC2 instances i.e. apache2 also runs on port 80.

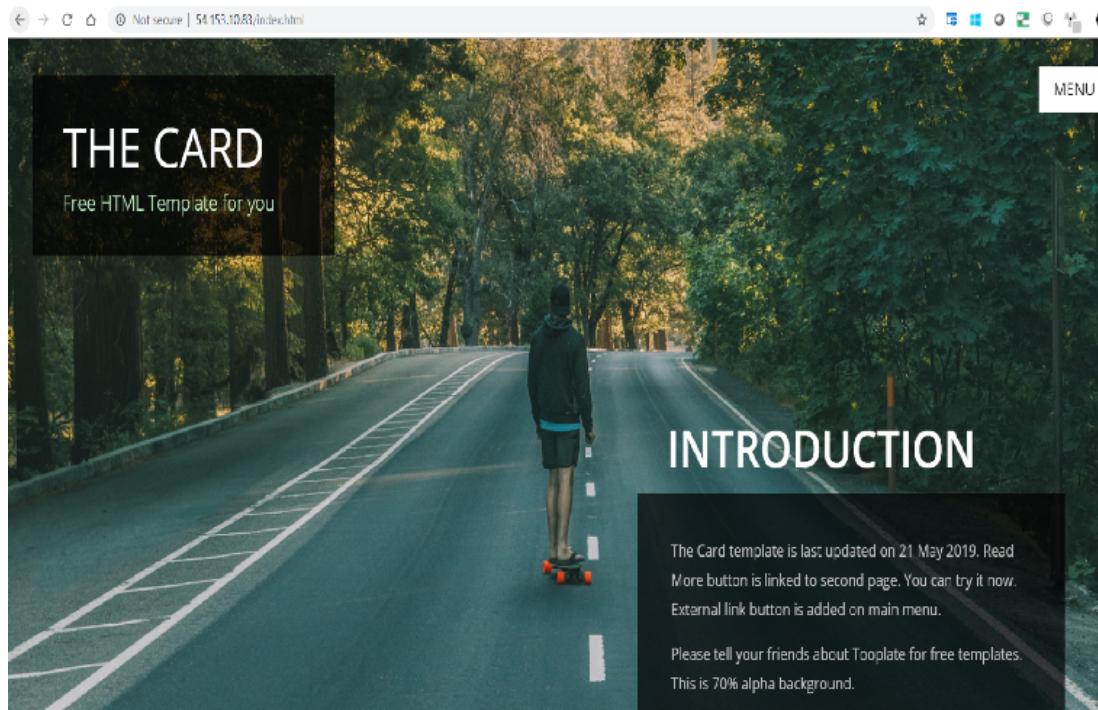
- Load Balancer Port is for front end port
- Instance Port is for backend port.

Next, we have to select the security group so here we have selected the same security group (card-web-elb-sg) which we have created for the load balancer in the article [Clustering Web Servers Using ELB](#).

The screenshot shows the 'Step 2: Assign Security Groups' wizard. Under 'Assign a security group', the 'Select an existing security group' radio button is selected. A table lists available security groups, with 'card-web-elb-sg' being selected.

Security Group ID	Name	Description	Actions
sg-01fe7610ef5dc05ba	card-web-elb-sg	card-web-elb-sg	Copy to new
sg-0886871d5dec0dd04	card-web-sg	card-web-sg	Copy to new
sg-38cde5e44	default	default VPC security group	Copy to new
sg-098951108a5650e43	Jenkins-sg	Jenkins-sg	Copy to new
sg-0b7039a27c42748a	Nexus-sg	Nexus-sg	Copy to new
sg-06201c172c97f9510	scriptbox-sg	This is used to run bash scripts	Copy to new

We can see that our existing EC2 instance card-web-01 is already up and running.



After the security group selection, we have to configure the health check , here in the below screenshot it says that load balancer is going to hit /index.html path to check if the EC2 instance is up or not and there are some other details like interval, response timeout etc. are also there which you can configure.

Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is a

Ping Protocol	HTTP
Ping Port	80
Ping Path	/index.html

Advanced Details

Response Timeout	5	seconds
Interval	30	seconds
Unhealthy threshold	2	
Healthy threshold	10	

Then we have to add the instances to the classic load balancer. As we can see that we have card-web01 running. So, we will add the same to the load balancer.

Step 5: Add EC2 Instances

The table below lists all your running EC2 Instances. Check the boxes in the Select column to add those instances to this load balancer.

VPC vpc-03aefae64 (172.31.0.0/16)

Instance	Name	State	Security groups
i-0a118c0e4017d4788	card-web01	running	card-web-sg
i-0bd45a30a7b0374f0	Jenkins-Server	stopped	Jenkins-sg
i-0713f468562f9298a	Nexus-Server	stopped	Nexus-sg
i-0cd00bf96357ba615	Scriptbox	stopped	scriptbox-sg

Availability Zone Distribution
1 instance in us-west-1b

Enable Cross-Zone Load Balancing (i)

Enable Connection Draining (i) 300 seconds

Last step is to add the tags and we have given the Name as card-classic-lb.

Step 6: Add Tags

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.

Key	Value
Name	Card-classic-lb

Create Tag

Finally review, click create and we can see that our classic load balancer has been created successfully.

Resource Groups

Create Load Balancer

Load balancer: card-web-prod-lb

Instances

Instance ID	Name	Availability Zone	Status	Actions
i-0a118c0e4017d4788	card-web01	us-west-1b	OutOfService <small>(i)</small>	Remove from Load Balancer

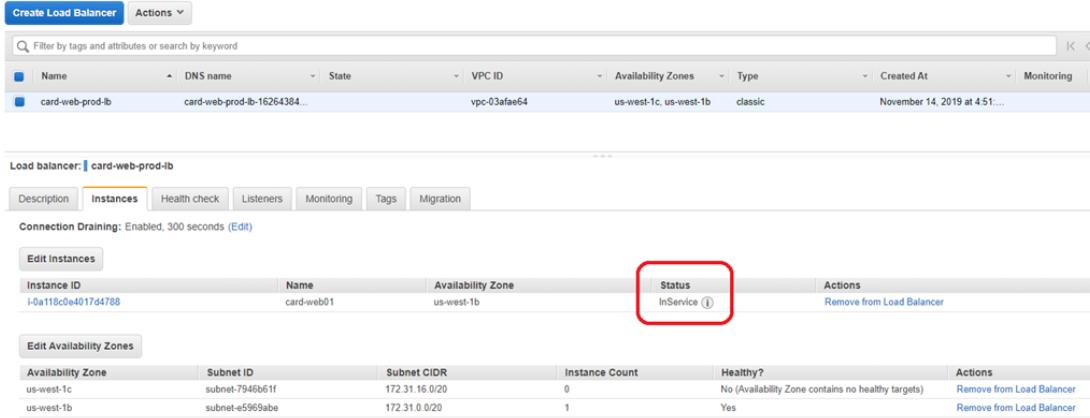
Edit Availability Zones

Availability Zone	Subnet ID	Subnet CIDR	Instance Count	Healthy?	Actions
us-west-1c	subnet-7946b61f	172.31.16.0/20	0	No (Availability Zone contains no healthy targets)	Remove from Load Balancer
us-west-1b	subnet-e5969abe	172.31.0.0/20	1	No (Availability Zone contains no healthy targets)	Remove from Load Balancer

And can see that the status of the EC2 instance is out of service.

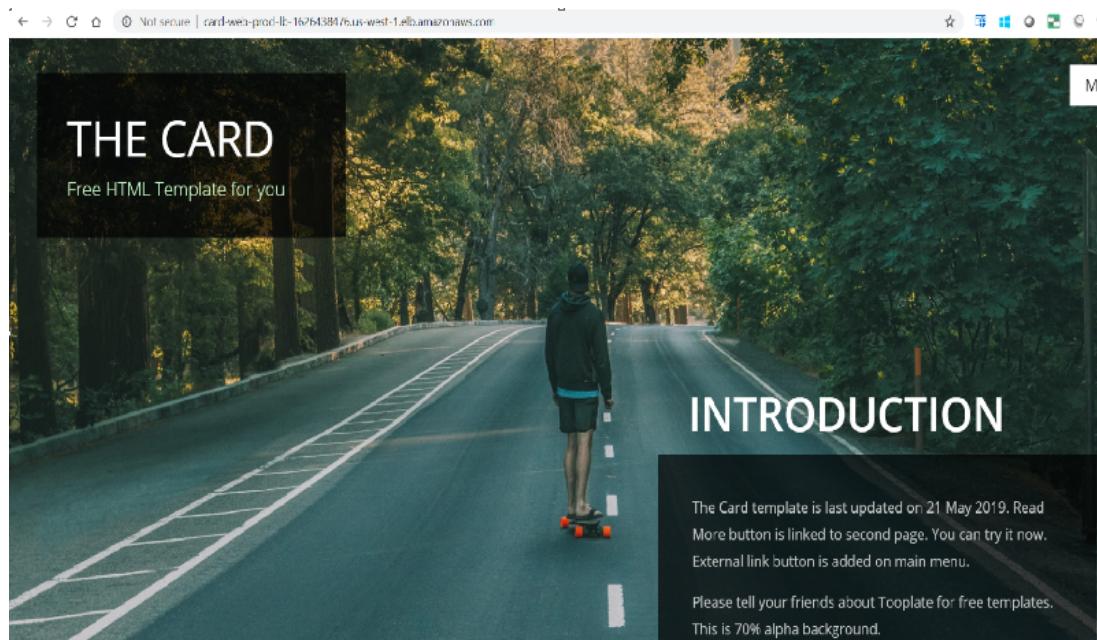
So, to make it work we will have to add/allow the load balancer security group (card-web-elb-sg) as an inbound rule of the security group of the EC2 instance (card-web01).

We can see that our newly added EC2 instance in the load balancer is in service.



The screenshot shows a line graph with 'CPU Utilization' on the Y-axis and 'Time' on the X-axis. The graph displays two data series: one for 'card-web-prod-lb' which remains consistently low (around 10-20%), and another for 'card-web-ami' which fluctuates between 50% and 80%.

We can also see that classic load balancer is also working and we have configured the EC2 instance in the load balancer correctly.



NEED FOR AUTO SCALING

Now suppose that we have a requirement wherein to meet the current application load we need 4 servers and if any one goes down , then we can re-create it using the AMI.

But what if 4 servers are not able to handle the load, then we will have to add more instances and we can gather this information by the cloudwatch alerts that whether the load is high or not.

But this process may take some time depending on when you observed that the servers are not able to handle the load, then you will have to create the other instances and this would be a time consuming process. What if all this can be automated??

With autoscaling we can scale out if the load is high or we can scale in or also delete the existing machines if the load is low and all that is possible with autoscaling.

AUTO SCALING



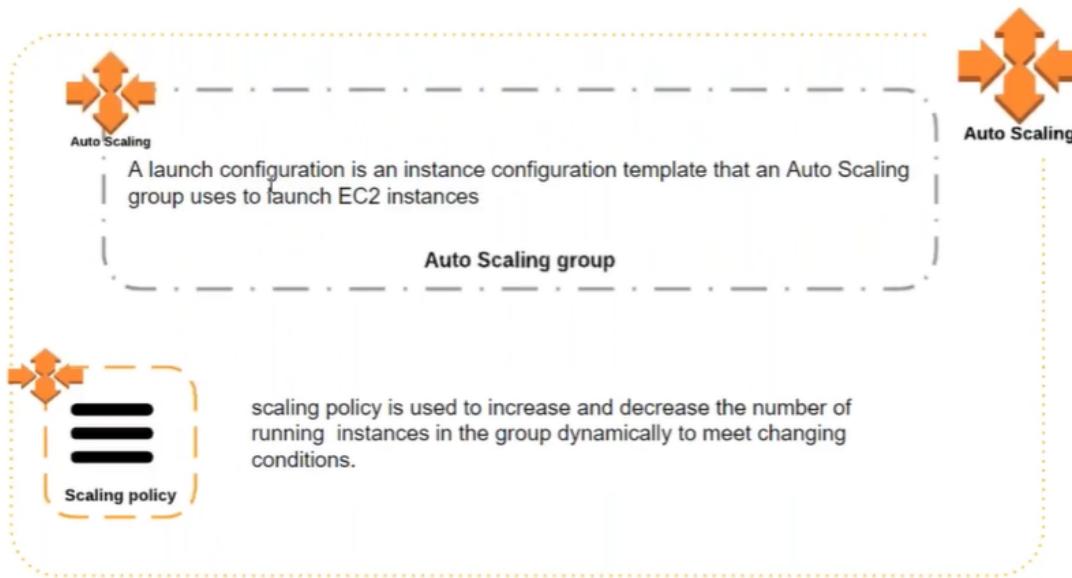
Auto Scaling is a service that automatically monitors and adjusts compute resources to maintain performance for applications hosted in the AWS.



Alarm monitors CloudWatch metrics for Instances.

We can set up alarm to monitor the servers and if alarm crosses the threshold then action could be like adding more instances or deleting some instances.

Auto Scaling is implemented using Launch Configuration and Scaling policies.



For example, we can tell the auto scaling group to have 4 instances always and can set up the auto scaling policies accordingly.

SETTING UP AUTO SCALING GROUP

Now let's get started with setting up the auto scaling group. For this we will be following the steps as mentioned below.

Go to the auto scaling service and click on create auto scaling group.

Welcome to Auto Scaling

You can use Auto Scaling to manage Amazon EC2 capacity automatically, maintain the right number of instances for your application, operate a healthy group of instances, and scale it according to your needs.

[Learn more](#)

[Create Auto Scaling group](#)

Note: To create your Auto Scaling groups in a different region, select your region from the navigation bar.

Benefits of Auto Scaling**Automated Provisioning**

Keep your Auto Scaling group healthy and balanced, whether you need one instance or 1,000.

[Learn more](#)

Adjustable Capacity

Maintain a fixed group size or adjust dynamically based on Amazon CloudWatch metrics.

[Learn more](#)

Launch Template Support

Provision instances easily using EC2 Launch Templates.

[Learn more](#)

Here we can choose the different options, but we will go with launch configuration.

Create Auto Scaling Group

Complete this wizard to create your Auto Scaling group. First, choose either a launch configuration or a launch template to specify the parameters that your Auto Scaling group uses to launch instances.

CREATING LAUNCH CONFIGURATION

To create it we Select the AMI which we created in the beginning and add other details as seen in the screenshot.

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

Create Launch Configuration

Name: card-web-LC

Purchasing option: Request Spot Instances

IAM role: None

Monitoring: Enable CloudWatch detailed monitoring

Advanced Details

Kernel ID: Use default

RAM Disk ID: Use default

User data:

(Optional)

IP Address Type:

- Only assign a public IP address to instances launched in the default VPC and subnet (default)
- Assign a public IP address to every instance.
- Do not assign a public IP address to any instances.

Note: this option only affects instances launched into an Amazon VPC

Later, if you want to use a different launch configuration, you can create a new one and apply it to any Auto Scaling group. Existing launch configurations cannot be edited.

Here if you want to execute some commands when the machine is up , you can do that in User Data.

Then click next till you get to choose the security group for the launch configuration.

1. Choose AMI 2. Choose Instance Type 3. Configure details 4. Add Storage 5. Configure Security Group 6. Review

Create Launch Configuration

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow traffic on the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group:

- Create a new security group
- Select an existing security group (sg-0888871d53ecbdd04 selected)

Security Group ID	Name	VPC ID	Description
sg-0d1e7610ef5dc5ba	card-web-elb-sg	vpc-03a9ae64	card-web-elb-sg
sg-0888871d53ecbdd04	card-web-sg	vpc-03a9ae64	card-web-sg

Inbound rules for sg-0888871d53ecbdd04 Selected security groups: sg-0888871d53ecbdd04

Type	Protocol	Port Range	Source
HTTP	TCP	80	sg-0d1e7610ef5dc5ba (card-web-elb-sg)
HTTP	TCP	80	61.95.254.10/32
SSH	TCP	22	61.95.254.10/32

We have selected the same security group as we did for the EC2 instance card-web01, because we want all our instances to have the same security group settings.

Then we will select the key pair we used for card-web01.

Select an existing key pair or create a new key pair

X

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair ▾

Select a key pair

card-kp ▾

I acknowledge that I have access to the selected private key file (card-kp.pem), and that without this file, I won't be able to log into my instance.

Cancel

Create launch configuration

CREATING AUTO SCALING GROUP

Now once we click on Create Launch Configuration , it will take you to the auto scaling group screen where we can add the subnet details (Now AWS may launch both 2 instances in one zone or 1 instance in each zone, it has its own mechanism). Select all the subnet which are available and then add the load balancer to handle the traffic.

Here in the below screenshot we can see that we have the launch configuration selected, group size given as 2 (means at any time we need 2 instances up and running) and we have also selected the classic load balancer.

aws Services Resource Groups

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

Group name: Card-web-ASG

Launch Configuration: card-web-LC

Group size: Start with instances

Network: vpc-03afae64 (172.31.0.0/16) (default)

Subnet: subnet-e5969abe(172.31.0.0/20) | Default in us-west-1b
subnet-7946b61f(172.31.16.0/20) | Default in us-west-1c

Create new subnet

Each instance in this Auto Scaling group will be assigned a public IP address.

Advanced Details

Load Balancing	<input type="checkbox"/> Receive traffic from one or more load balancers	Learn about Elastic Load Balancing
Classic Load Balancers	<input type="text" value="card-web-prod-lb"/>	
Target Groups	<input type="text"/>	
Health Check Type	<input checked="" type="radio"/> ELB <input type="radio"/> EC2	
Health Check Grace Period	<input type="text" value="300"/> seconds	
Monitoring	Amazon EC2 Detailed Monitoring metrics, which are provided at 1 minute frequency, are not enabled for the launch configuration card-web-LC. Instances launched from it will use Basic Monitoring metrics, provided at 5 minute frequency.	
Learn more		
Instance Protection	<input type="text"/>	
Service-Linked Role	<input type="text" value="AWSServiceRoleForAutoScaling"/> C	
The default role does not exist. It will be automatically created on your behalf.		

Here we are doing health check on ELB rather than EC2 because our ELB is already doing the health check on the EC2 instance.

NOTE

We never store data in the volumes of the instances in the auto scaling group as they get deleted and recreated many times. So, if we really want to store some data then we can do it in the S3 storage service.

CHOOSING SCALING POLICIES

As we need to scale out and scale in depending on the load, so we will select the scaling policies as shown below.

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

You can optionally add scaling policies if you want to adjust the size (number of instances) of your group automatically. A scaling policy is a set of instructions for making such adjustments in response to a specific event. You can remove a scaling policy at any time. When the alarm triggers, it will execute the policy and adjust the size of your group.

Keep this group at its initial size
 Use scaling policies to adjust the capacity of this group

Scale between and instances. These will be the minimum and maximum size of your group.

Increase Group Size

Name:	<input type="text" value="Increase Group Size"/>
Execute policy when:	No alarm selected C Add new alarm
Take the action:	Add <input type="text" value="0"/> instances Add step
Instances need:	<input type="text" value="300"/> seconds to warm up after each step
Create a simple scaling policy	

Decrease Group Size

Name:	<input type="text" value="Decrease Group Size"/>
Execute policy when:	No alarm selected C Add new alarm
Take the action:	Remove <input type="text" value="0"/> instances Add step

Here we have added that we need min 2 and max 10 instances possible. Now we need to create alarm and depending on the alarm we will be doing some actions. So, click on add new alarm.

X

Create Alarm

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

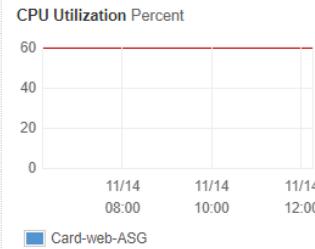
Send a notification to: CARD-WEB01-ALARM (tyagiamit08@gr [create topic](#)

Whenever: Average of CPU Utilization

Is: \geq 60 Percent

For at least: 1 consecutive period(s) of 5 Minutes

Name of alarm: awsec2-Card-web-ASG-CPU-Utilization



Cancel

Create Alarm

So above it says that if the CPU utilization is $> 60\%$ for consecutive 5 mins then raise this alarm. We will also add alarm for the decreasing group size as if the CPU utilization is $< 40\%$ for consecutive 5 mins then raise this alarm.

Edit Alarm

You can use CloudWatch alarms to be notified automatically whenever metric data reaches a level you define.

To edit an alarm, first choose whom to notify and then define when the notification should be sent.

Send a notification to: CARD-WEB01-ALARM (tyagiamit08@gr [create topic](#)

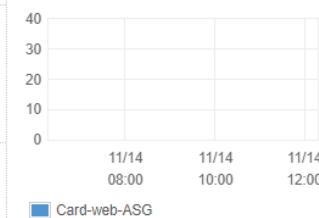
Whenever: Average of CPU Utilization

Is: \leq 40 Percent

For at least: 1 consecutive period(s) of 5 Minutes

Name of alarm: awsec2-Card-web-ASG-High-CPU-Utilization

CPU Utilization Percent



Cancel

Save

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

Scale between **1** and **10** instances. These will be the minimum and maximum size of your group.

Increase Group Size

Name: **Increase Group Size**

Execute policy when: awsec2-Card-web-ASG-CPU-Utilization [Edit](#) [Remove](#)
breaches the alarm threshold: CPUUtilization >= 60 for 300 seconds
 for the metric dimensions AutoScalingGroupName = Card-web-ASG

Take the action:

- Add **1** instances when **60** <= CPUUtilization < **70**
- Add **2** instances when **70** <= CPUUtilization < **+infinity**

[Add step](#)

Instances need: **60** seconds to warm up after each step

[Create a simple scaling policy](#)

Decrease Group Size

Name: **Decrease Group Size**

Execute policy when: awsec2-Card-web-ASG-High-CPU-Utilization [Edit](#) [Remove](#)
breaches the alarm threshold: CPUUtilization <= 40 for 300 seconds
 for the metric dimensions AutoScalingGroupName = Card-web-ASG

Take the action:

- Remove **1** instances when **40** >= CPUUtilization > **30**
- Remove **2** instances when **30** >= CPUUtilization > **-infinity**

[Add step](#)

[Create a simple scaling policy](#)

So finally, we can see the complete scaling policies as shown above. Then next is to configure notification, which means when we want the team to be notified.

aws Services Resource Groups *

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

Configure your Auto Scaling group to send notifications to a specified endpoint, such as an email address, whenever a specified event takes place, including: successful launch of an instance

If you created a new topic, check your email for a confirmation message and click the included link to confirm your subscription. Notifications can only be sent to confirmed addresses.

Send a notification to: **CARD-WEB01-ALARM (tyagamit08@gmail.com)** [create topic](#) [X](#)

Whenever instances:

- launch
- terminate
- fail to launch
- fail to terminate

[Add notification](#)

Then add tags if required.

aws Services Resource Groups *

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

A tag consists of a case sensitive key-value pair that you can use to identify your group. For example, you could define a tag with Key = Environment and Value = Production. You can optionally choo:

Key	Value
Name	Card-web-asg

[Add tag](#) 49 remaining

Finally review and click on Create and we can see that our auto scaling group has been created successfully.

Auto Scaling Group: Card-web-ASG

Details Activity History Scaling Policies Instances Monitoring Notifications Tags Scheduled Actions Lifecycle Hooks

Launch Configuration: card-web-LC

Desired Capacity: 2

Min: 2

Max: 10

Availability Zone(s): us-west-1b, us-west-1c

Subnet(s): subnet-e5969abe, subnet-7946b61f

Classic Load Balancers: card-web-prod-lb

Target Groups

Health Check Type: EC2

Health Check Grace Period: 300

Instance Protection

Termination Policies: Default

NOTE

We can see that it shows 0 instances because it does not consider your old instances or the instances which are part of the Load Balancer previously before creation of auto scaling group.

So, first thing it will do is , it will launch the instances and we can see that in the activity history as well.

Auto Scaling Group: Card-web-ASG

Details Activity History Scaling Policies Instances Monitoring Notifications Tags Scheduled Actions Lifecycle Hooks

Instances

Instance ID	Lifecycle	Launch Configuration / Template	Availability Zone	Health Status
i-0ddd6c5599217a056	Pending	card-web-LC	us-west-1b	Healthy
i-0fc5c764e1aeb644a	Pending	card-web-LC	us-west-1c	Healthy

Auto Scaling Group: Card-web-ASG

Details Activity History Scaling Policies Instances Monitoring Notifications Tags Scheduled Actions Lifecycle Hooks

Activity History

Status	Description	Start Time
Successful	Launching a new EC2 instance: i-0fc5c764e1aeb644a	2019 November 14 17:52:39 UTC+5:30
Successful	Launching a new EC2 instance: i-0ddd6c5599217a056	2019 November 14 17:52:39 UTC+5:30
Failed	Launching a new EC2 instance	2019 November 14 17:50:39 UTC+5:30
Failed	Launching a new EC2 instance	2019 November 14 17:50:38 UTC+5:30

Auto Scaling Group: Card-web-ASG

Details Activity History Scaling Policies Instances Monitoring Notifications Tags Scheduled Actions Lifecycle Hooks

Instances

Instance ID	Lifecycle	Launch Configuration / Template	Availability Zone	Health Status
i-0ddd6c5599217a056	InService	card-web-LC	us-west-1b	Healthy
i-0fc5c764e1aeb644a	InService	card-web-LC	us-west-1c	Healthy

So, the newly launched instances are in service and are healthy and in the instance section we can see the 2 newly launched instances too.

As the previous instance is not part of the auto scaling group so we can terminate that as well.

NOTE

Auto Scaling group also adds the new instances in the load balancer .

We can see that the 2 newly created instances have also become part of the load balance and are in in service.

DELETING INSTANCE TO CHECK AUTO SCALING

Now as we have done the setup, so let's check if things are working as expected.

For this we are going to manually delete one instance and see if auto scaling group creates one more instance or not as we have given the desired no of instances as 2, so Ideally it should launch one more instance.

The screenshot shows the AWS Auto Scaling Groups page. A search bar at the top has 'Filter: Card-web-ASG'. Below it is a table with one row for 'Card-web-ASG'.

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grace Period
Card-web-ASG	card-web-LC	2	2	2	10	us-west-1b, us-west-1c	300	300

Below the table, the 'Auto Scaling Group: Card-web-ASG' section is expanded. It shows the following configuration:

- Launch Configuration:** card-web-LC
- Desired Capacity:** 2
- Min:** 2
- Max:** 10
- Availability Zone(s):** us-west-1b, us-west-1c
- Subnet(s):** subnet-e5969abe, subnet-7946b61f
- Classic Load Balancers:** card-web-prod-lb
- Target Groups:**
- Health Check Type:** EC2
- Health Check Grace Period:** 300

Here we can see that there are 2 instances attached to the auto scaling group. So, if I delete one then auto scaling group will create another automatically.

As you can see that we have triggered the termination process on one of the instances.

The screenshot shows the AWS Instances page. A search bar at the top has 'search: card'. Below it is a table with several rows of instance data, including the Card-web-ASG instance which is currently shutting down.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name
Card-web-ASG	i-0387ed4a5785219cc	t2.micro	us-west-1b	shutting-down	None	None	ec2-13-52-242-234.us-west-1.compute.amazonaws.com	13.52.242.234	-	card-4
card-web01	i-0a118c0e4017d4788	t2.micro	us-west-1b	running	2/2 checks ...	OK	ec2-54-183-145-211.us-west-1.compute.amazonaws.com	54.183.145.211	-	card-3
Card-web-ASG	i-0db2dc59a59b1d9ff	t2.micro	us-west-1c	running	2/2 checks ...	None	ec2-54-215-220-87.us-west-1.compute.amazonaws.com	54.215.220.87	-	card-4
Card-web-ASG	i-02c49a37234c01a58	t2.micro	us-west-1b	terminated	None	None	-	-	-	card-4
Card-web-ASG	i-0648f0618a6cf2a35	t2.micro	us-west-1c	terminated	None	None	-	-	-	card-4

Below the table, the details for the instance i-0387ed4a5785219cc are shown. The 'Status Checks' tab is selected, showing the instance is currently shutting down.

And if we go to the auto scaling group section , we can check in the activity history that it is launching one new instance.

The screenshot shows the AWS Auto Scaling Groups page. A search bar at the top has 'Filter: Card-web-ASG'. Below it is a table with one row for 'Card-web-ASG'.

Name	Launch Configuration	Instances	Desired	Min	Max	Availability Zones	Default Cooldown	Health Check Grace Period
Card-web-ASG	card-web-LC	2	2	2	10	us-west-1b, us-west-1c	300	300

Below the table, the 'Auto Scaling Group: Card-web-ASG' section is expanded. It shows the 'Activity History' tab selected. The history table shows three entries:

Status	Description	Start Time	End Time
Not yet in service	Launching a new EC2 instance: i-0baa44c27a10846d8	2019 November 15 17:28:12 UTC+5:30	
Successful	Terminating EC2 instance: i-0387ed4a5785219cc	2019 November 15 17:27:40 UTC+5:30	2019 November 15 17:27:48 UTC+5:30
Successful	Launching a new EC2 instance: i-0db2dc59a59b1d9ff	2019 November 15 17:18:14 UTC+5:30	2019 November 15 17:18:46 UTC+5:30

And again, we have 2 instances running.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
card-web01	i-0a118c0e4017d4788	t2.micro	us-west-1b	running	2/2 checks ... OK	ec2-54-183-145-211.us...	
Card-web-asg	i-0baa44c27a10846d8	t2.micro	us-west-1b	running	Initializing	None	ec2-54-153-25-202.us...
Card-web-asg	i-0db2dc99a69b1d9ff	t2.micro	us-west-1c	running	2/2 checks ... None	ec2-54-215-220-87.us...	

Great 😊 so auto scaling works as expected. We can also see how auto scaling group creates or terminates different instances depending on the alarms and actions we have set up.

HOW DO YOU UPGRADE YOUR INSTANCES WHICH ARE PART OF AUTO SCALING GROUP??

Now one very interesting questions would be coming to your mind, that how we can update / upgrade the EC2 instances which part of the auto scaling group. To do so, we will have to follow the below steps,

- First create a new AMI which will have the changes/upgrades.
- Then create new launch configuration.
- Then update newly created launch configuration name in the auto scaling group.
- Then delete all the instances one by one or all at once depending on the feasibility.
- Finally, auto scaling group will automatically launch the new instances from the new launch configuration.

So, in this way we can easily update/upgrade the instances in the auto scaling group. Every change in auto scaling group is done through AMI and launch configuration.

SUMMARY

In this article we have covered a lot of topics like Introduction to auto scaling, why do we need it, its practical implementation, classic load balancer, use of AMI etc. We also saw how we can practically set up auto scaling group for our applications with launch configuration and different scaling policies.

I hope you find this article helpful. Stay tuned for more ... Cheers!!

Thank you for using C# Corner