## 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.

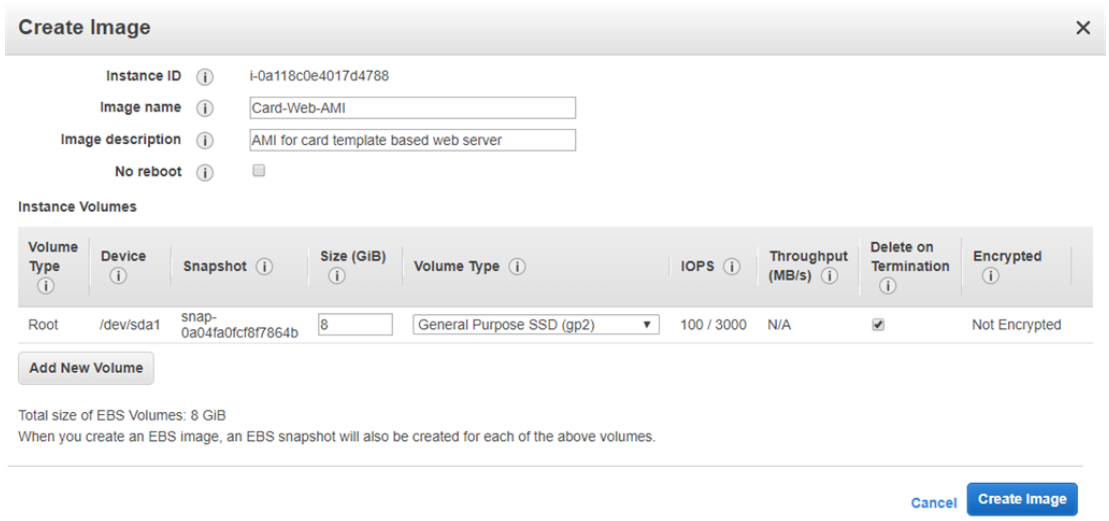
## CREATING AMI FOR EXISTING EC2 INSTANCE

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.

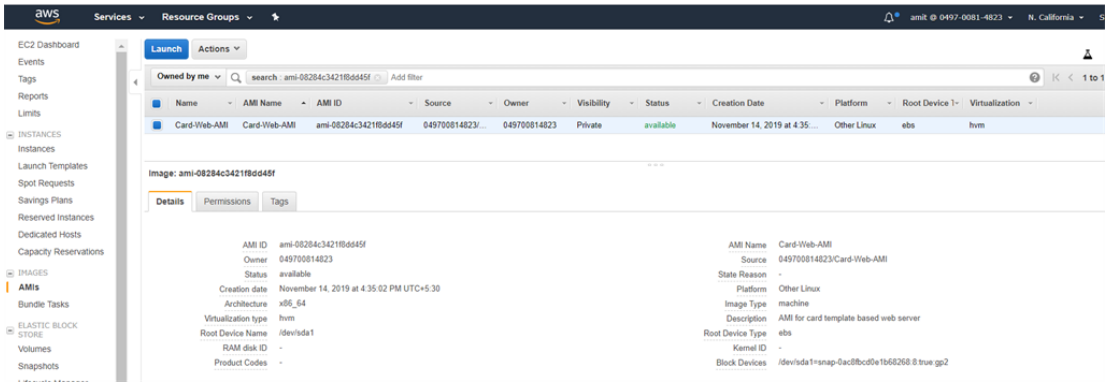
In one of the previous article Clustering Web Servers Using ELB we have seen how we can create the AMI from an existing EC2 instance.

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.

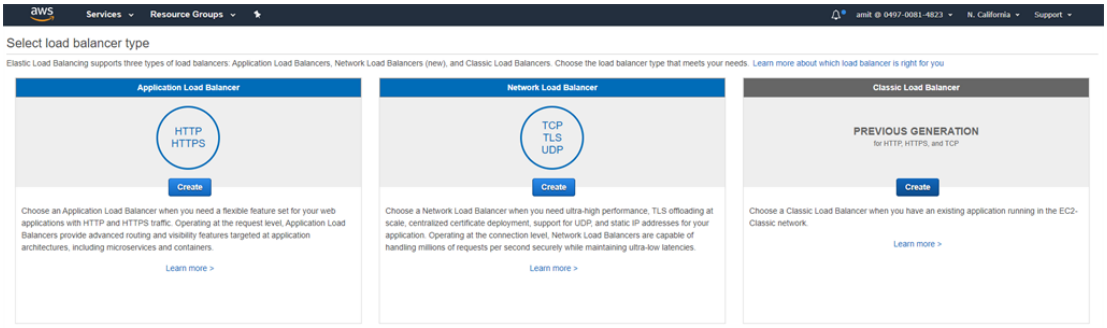



# CREATING CLASSIC LOAD BALANCER

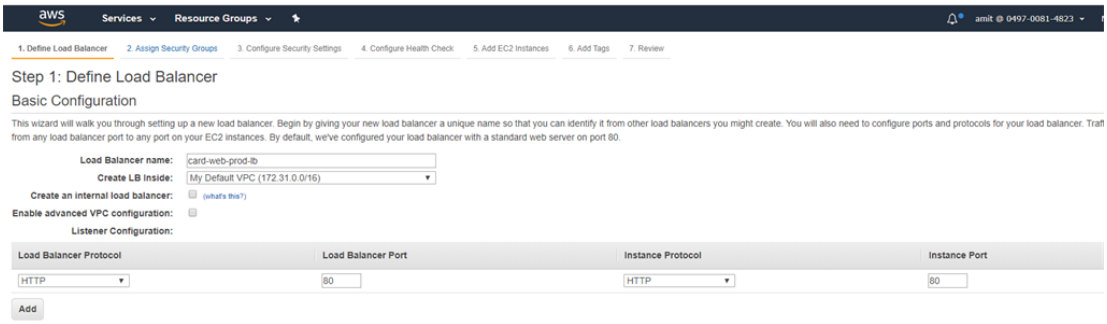
Now we have our AMI ready which we named as Card-Web-AMI and has one EC2 instance card-web01. So next we have to create the load balancer to balance the load. In this article we are going to create the Classic Load Balancer. The *Classic Load Balancer* operates on both the request and connection levels. 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.



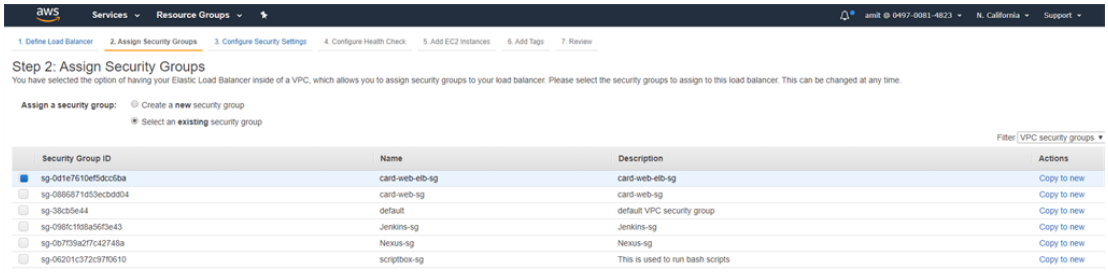

Here we have selected the classic Load balancer and has given it the name as card-web-prod-lb.



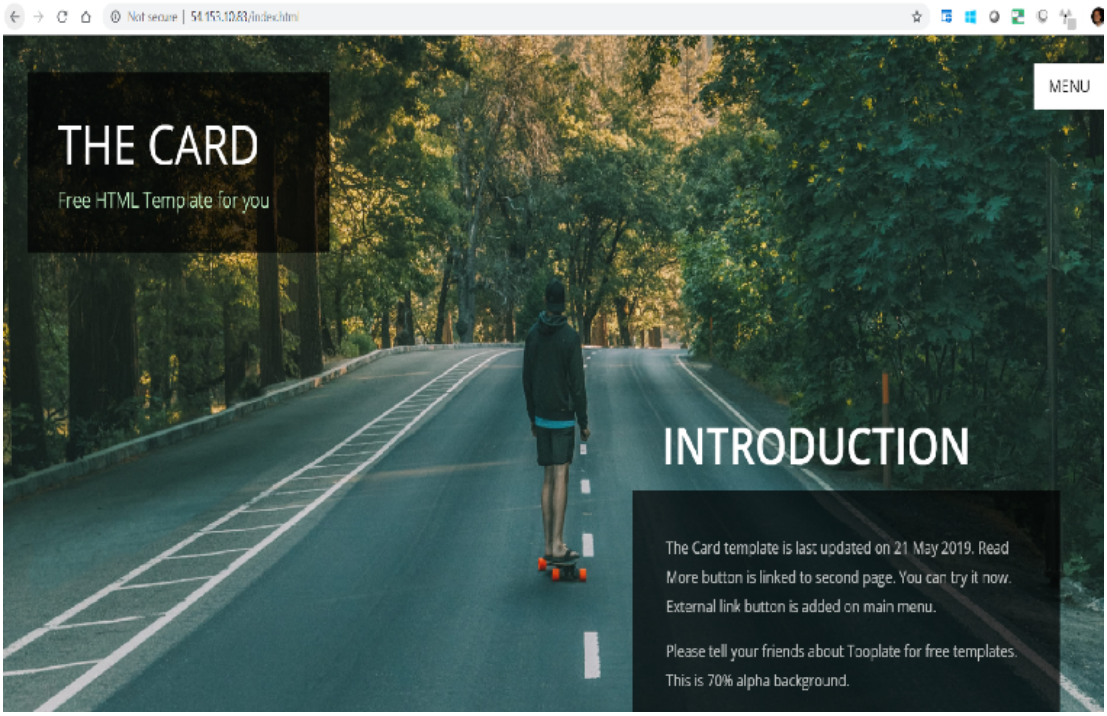

We have also given the front end and back end port as 80 as our HTTP request comes on port 80 and the service running in EC2 instances i.e. apache2 also runs on port 80.

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

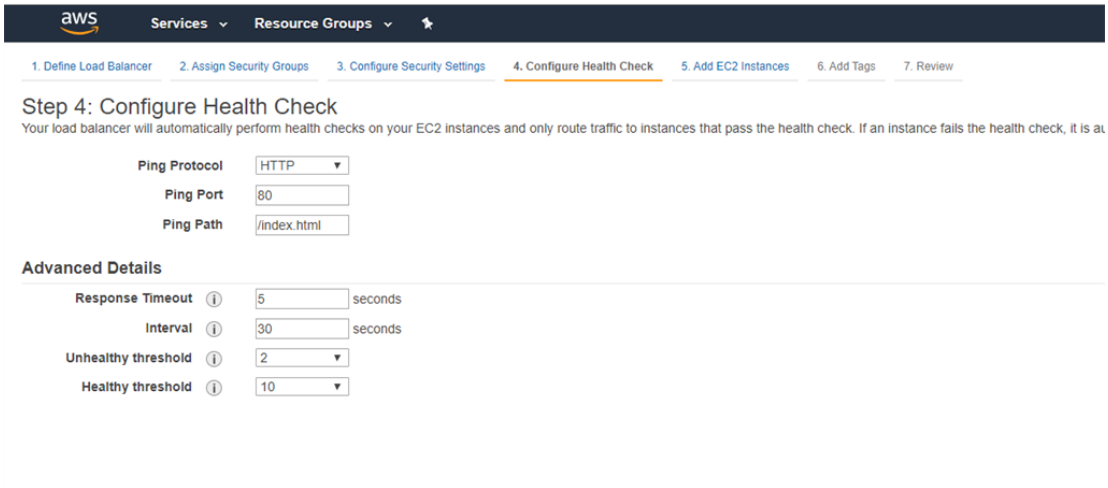
Next, we have to select the security group so here we have selected the same security group (card-web-elb-sg ) which we have created for the load balancer in the article Clustering Web Servers Using ELB.



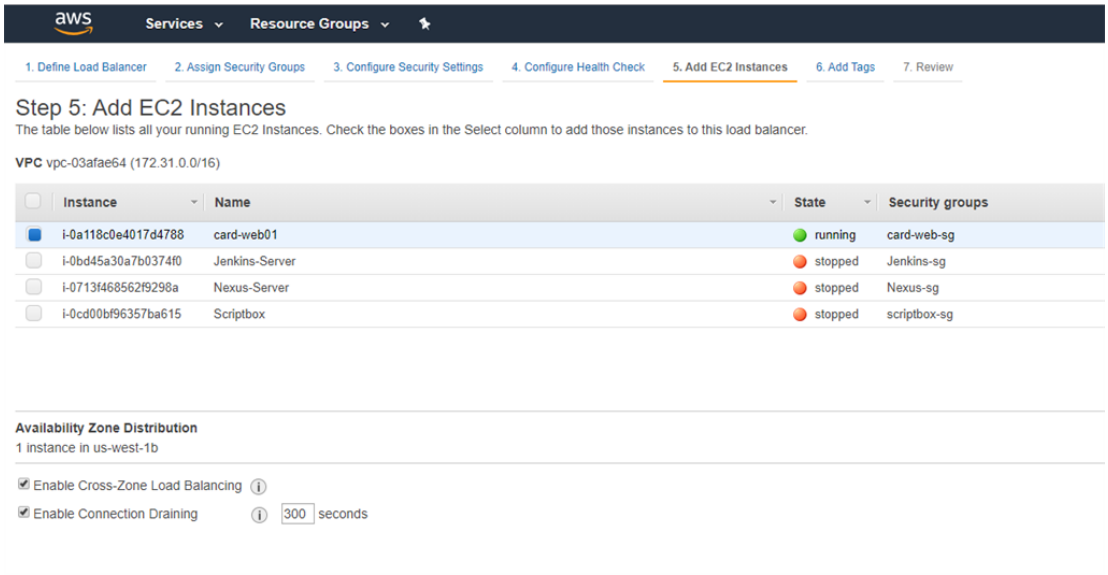

We can see that our existing EC2 instance card-web-01 is already up and running.



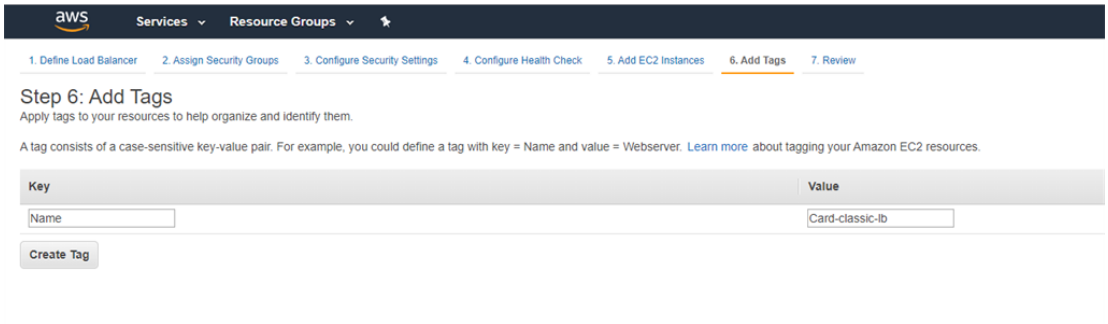

After the security group selection, we have to configure the health check , here in the below screenshot it says that load balancer is going to hit /index.html path to check if the EC2 instance is up or not and there are some other details like interval, response timeout etc. are also there which you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



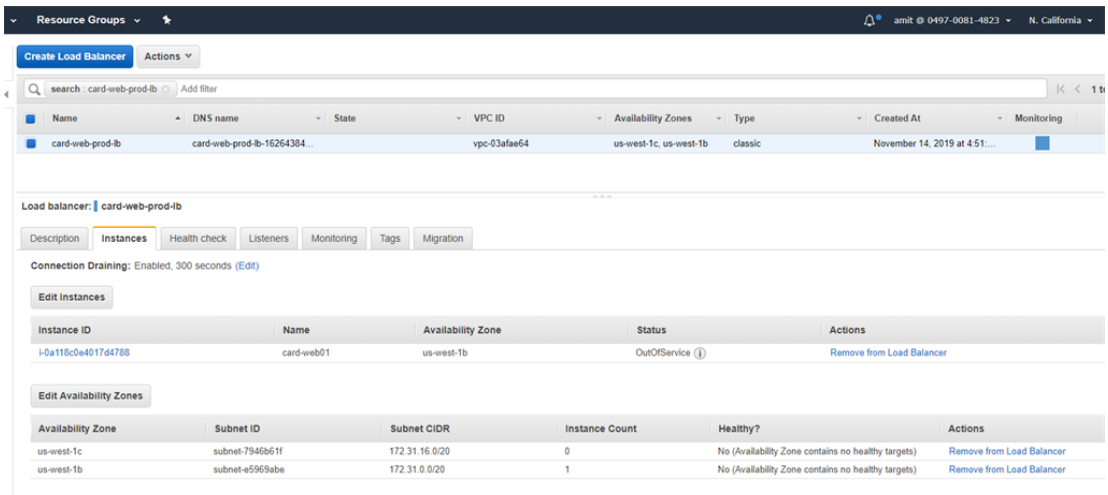

Then we have to add the instances to the classic load balancer. As we can see that we have card-web01 running. So, we will add the same to the load balancer.




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



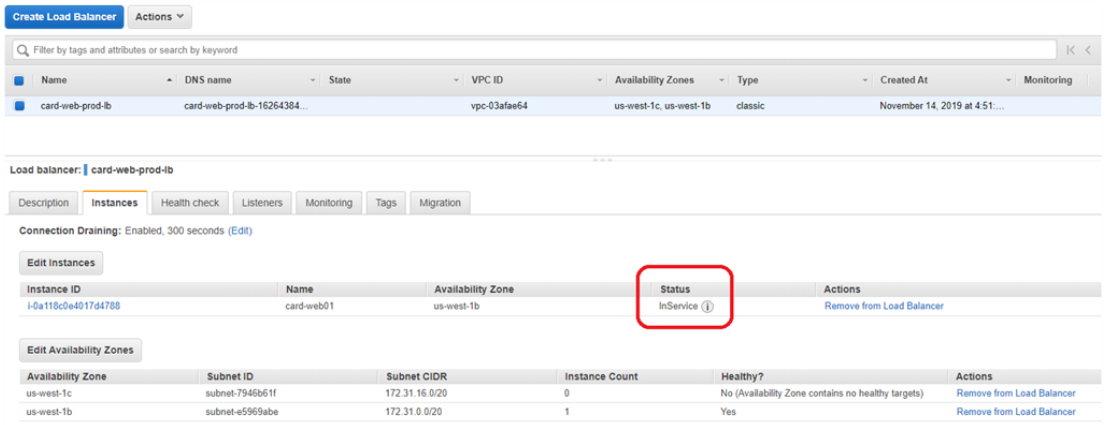

Finally review, click create and we can see that our classic load balancer has been created successfully.



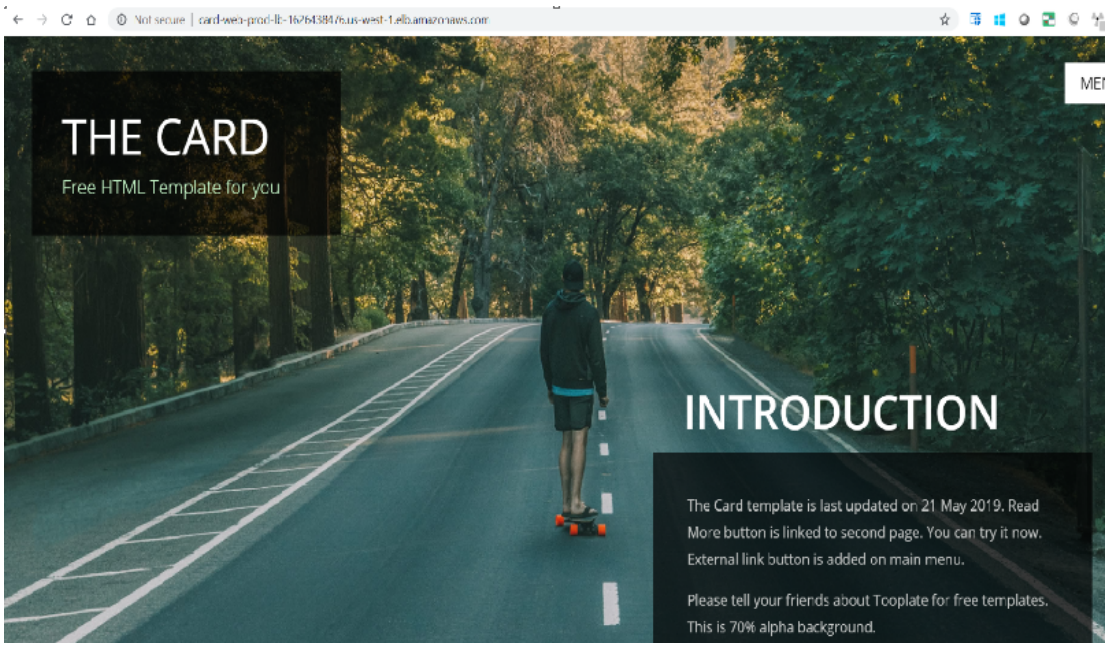

And can see that the status of the EC2 instance is out of service.

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.




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



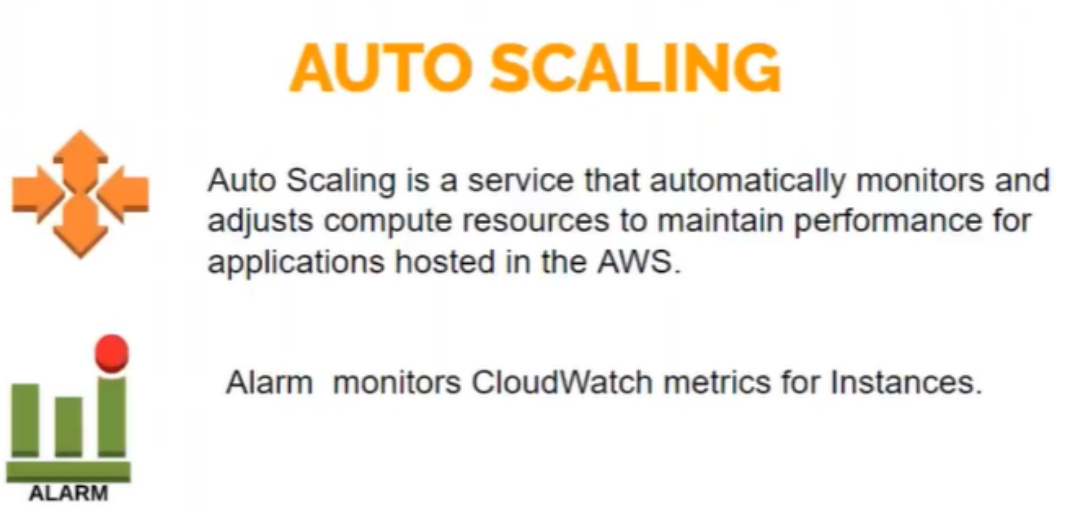

# NEED FOR AUTO SCALING

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.

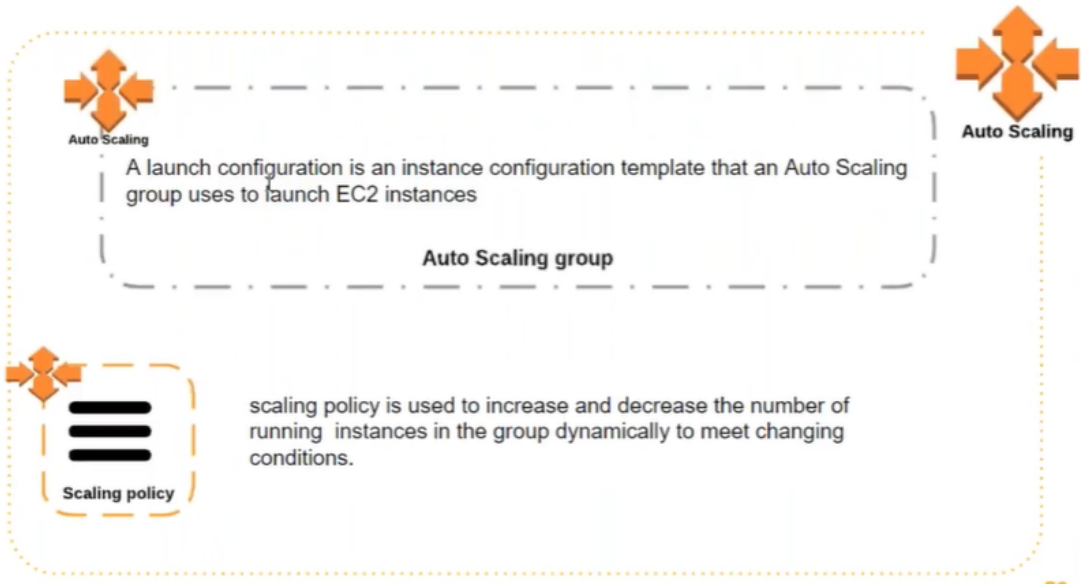
But this process may take some time depending on when you observed that the servers are not able to handle the load, then you will have to create the other instances and this would be a time consuming process. What if all this can be automated??

With autoscaling we can scale out if the load is high or we can scale in or also delete the exis7ing machines if the load is low and all that is possible with autoscaling.




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.

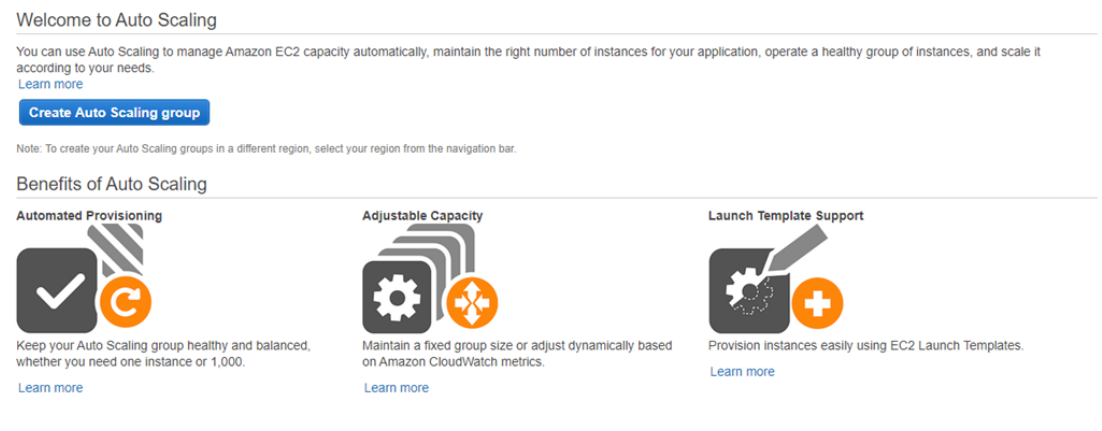



For example, we can tell the auto scaling group to have 4 instances always and can set up the auto scaling policies accordingly.

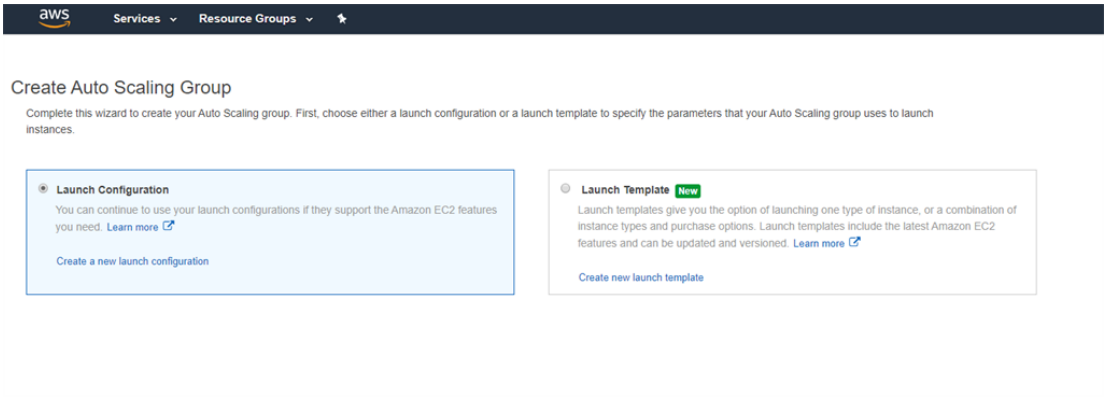
## SETTING UP AUTO SCALING GROUP

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.

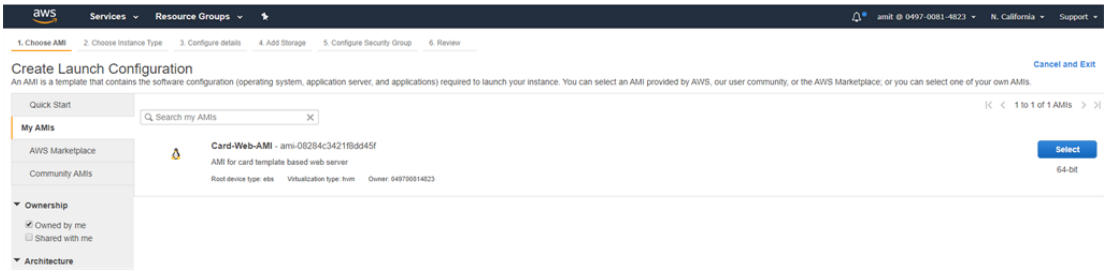


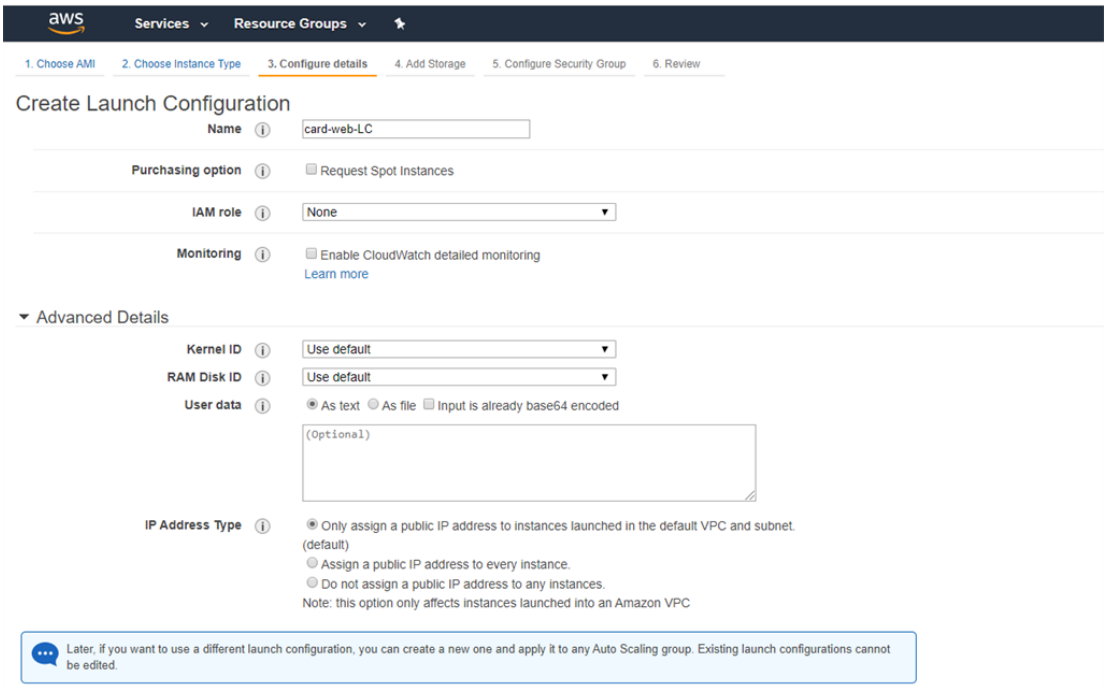

Here we can choose the different options, but we will go with launch configuration.




# CREATING LAUNCH CONFIGURATION

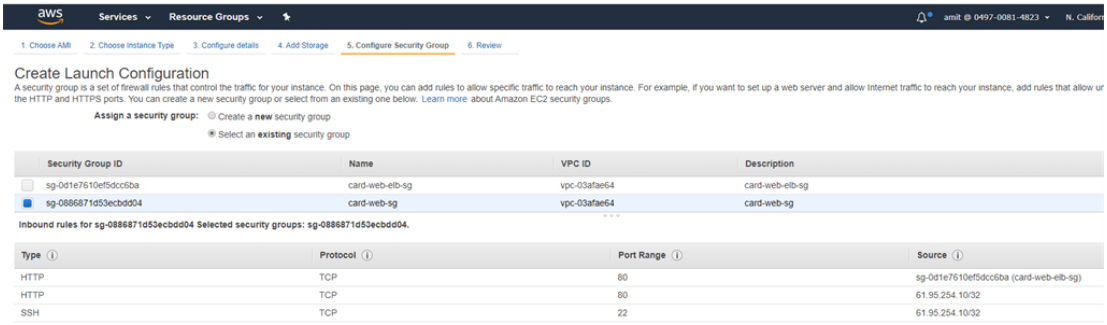
To create it we Select the AMI which we created in the beginning and add other details as seen in the screenshot.





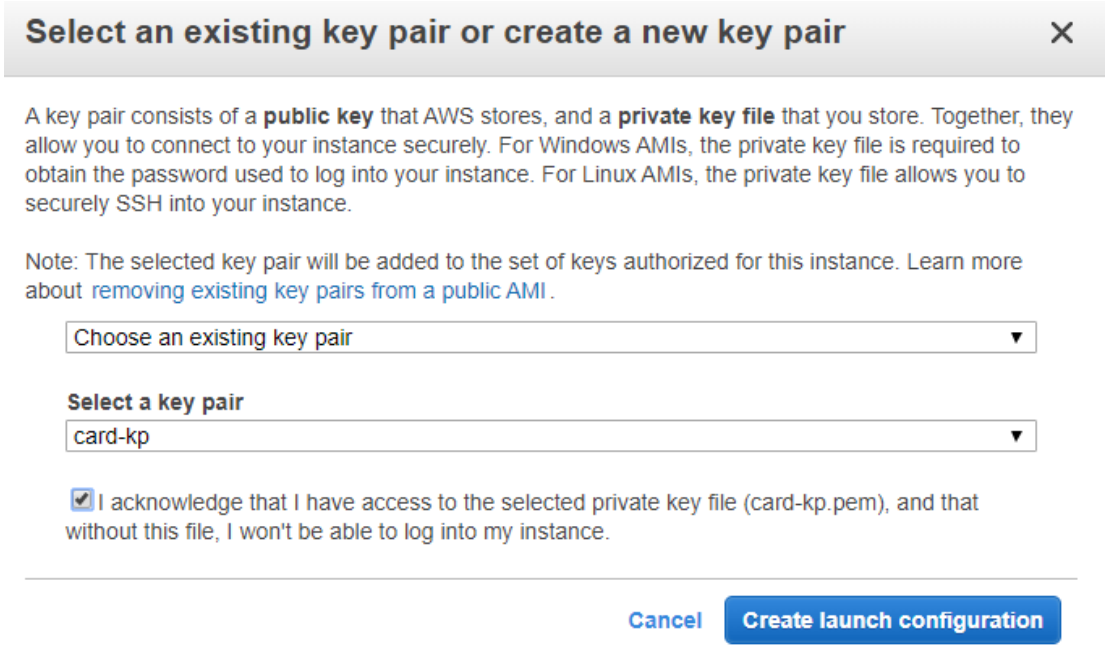

Here if you want to execute some commands when the machine is up , you can do that in User Data.

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




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.

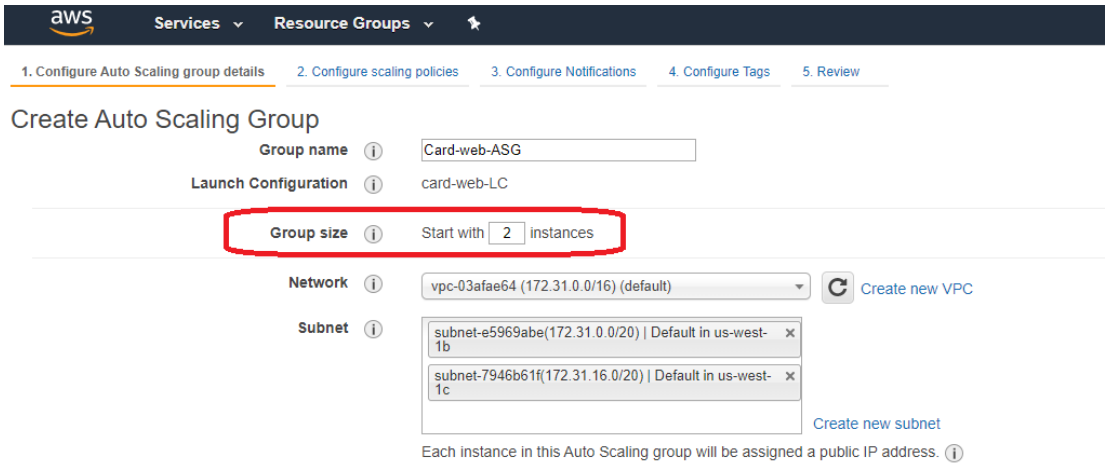
Then we will select the key pair we used for card-web01.

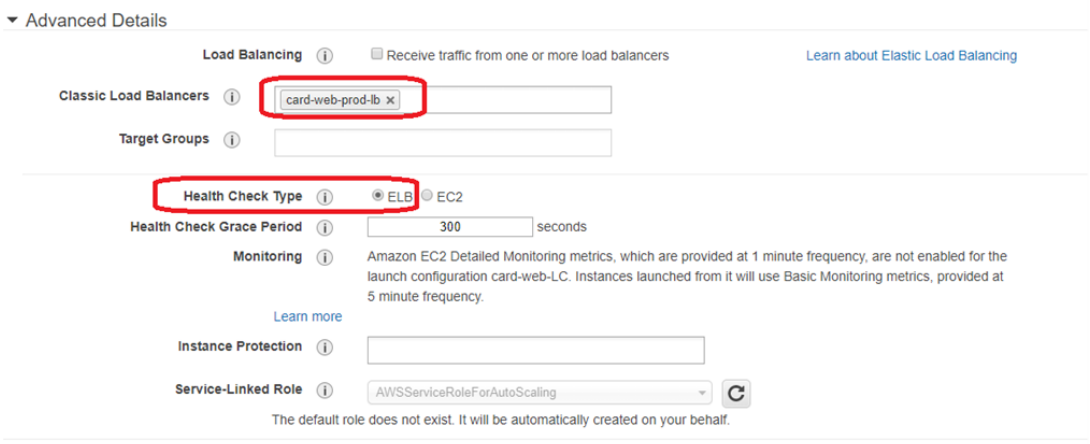



# CREATING AUTO SCALING GROUP

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.



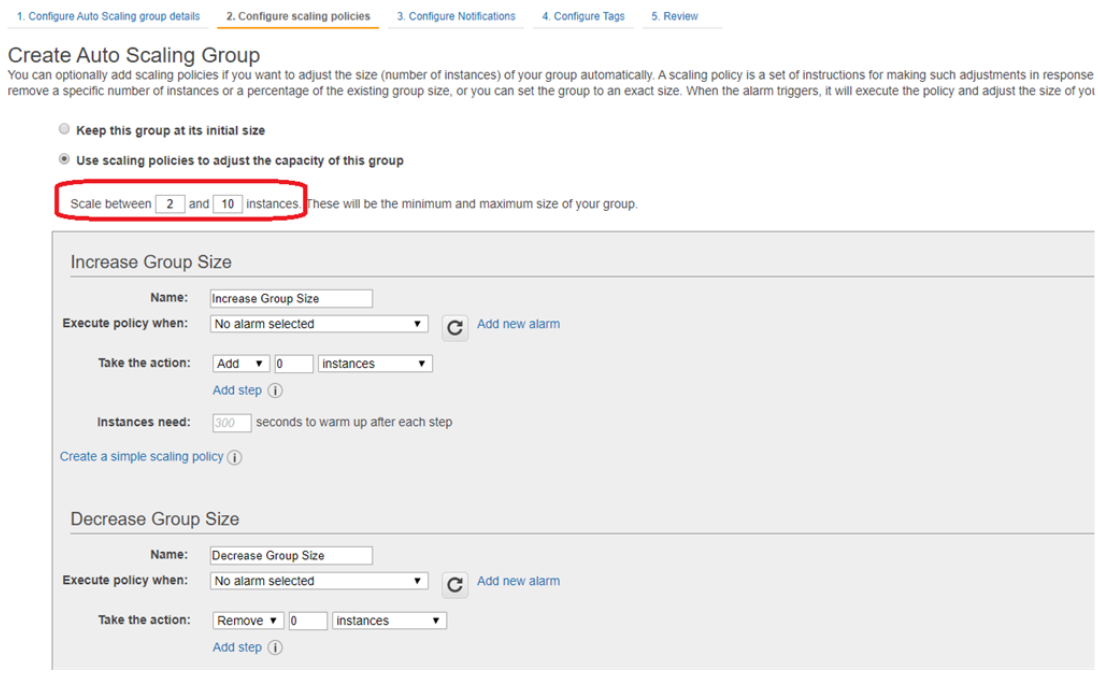



Here we are doing health check on ELB rather than EC2 because our ELB is already doing the health check on the EC2 instance.

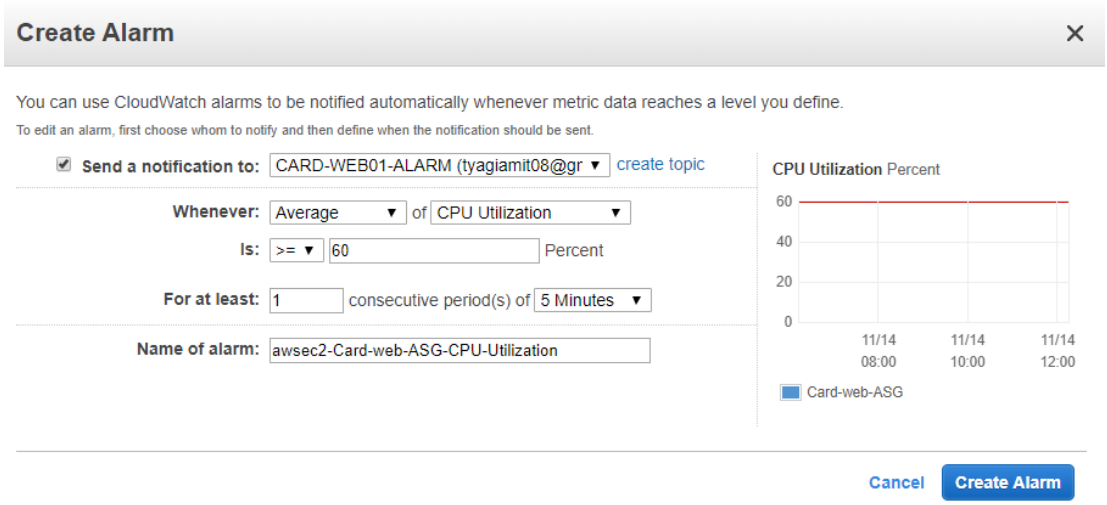
**NOTE**
We never store data in the volumes of the instances in the auto scaling group as they get deleted and recreated many times. So, if we really want to store some data then we can do it in the S3 storage service.

# CHOOSING SCALING POLICIES

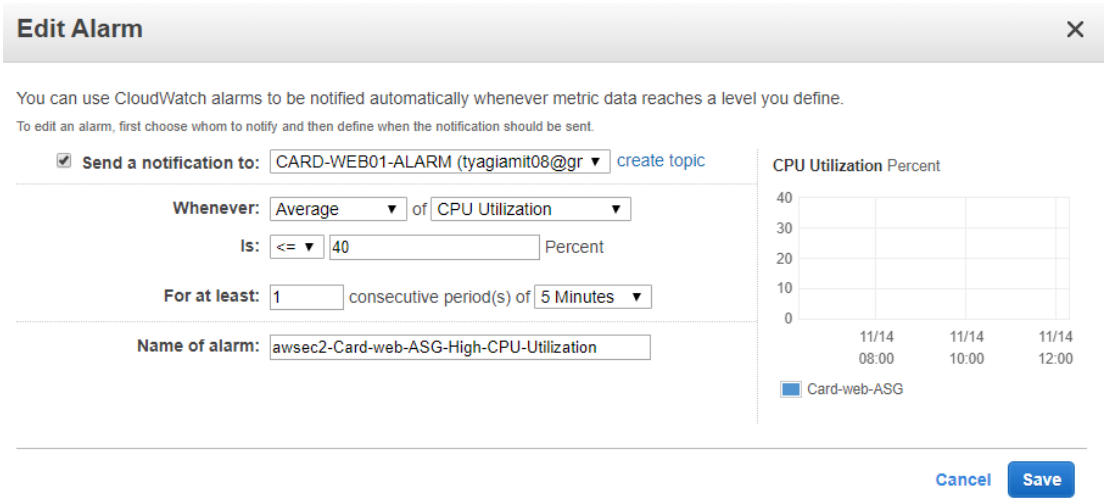
As we need to scale out and scale in depending on the load, so we will select the scaling policies as shown below.

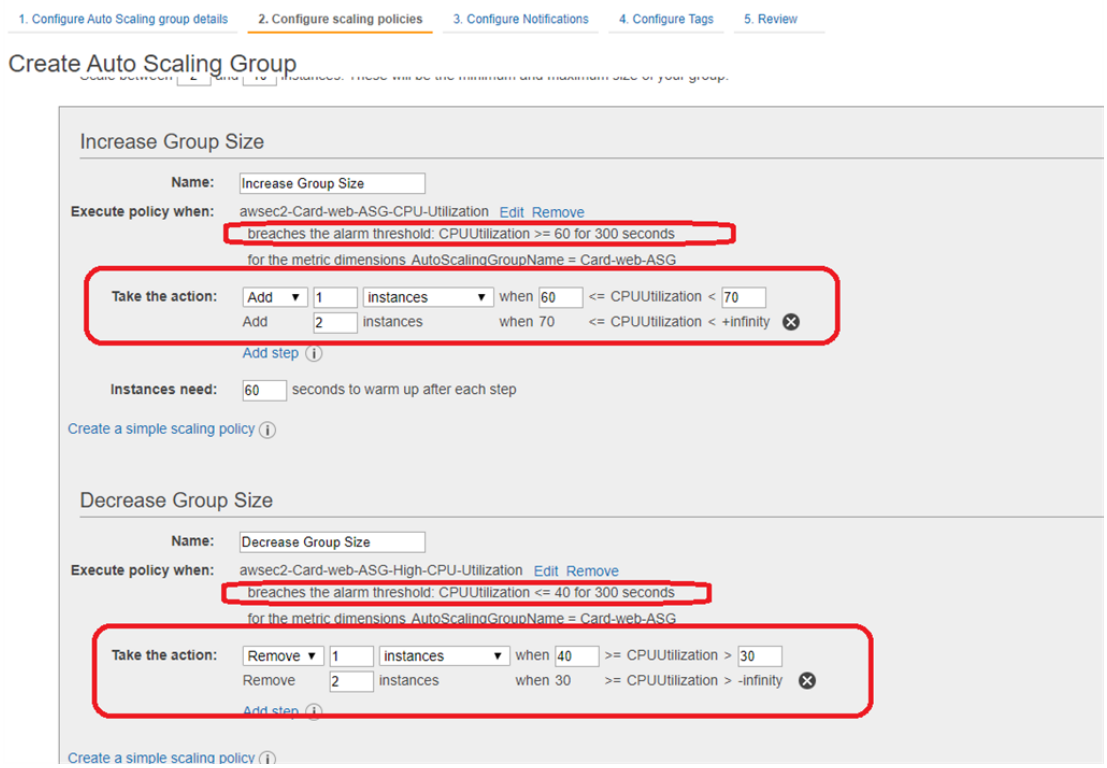



Here we have added that we need min 2 and max 10 instances possible. Now we need to create alarm and depending on the alarm we will be doing some actions. So, click on add new alarm.

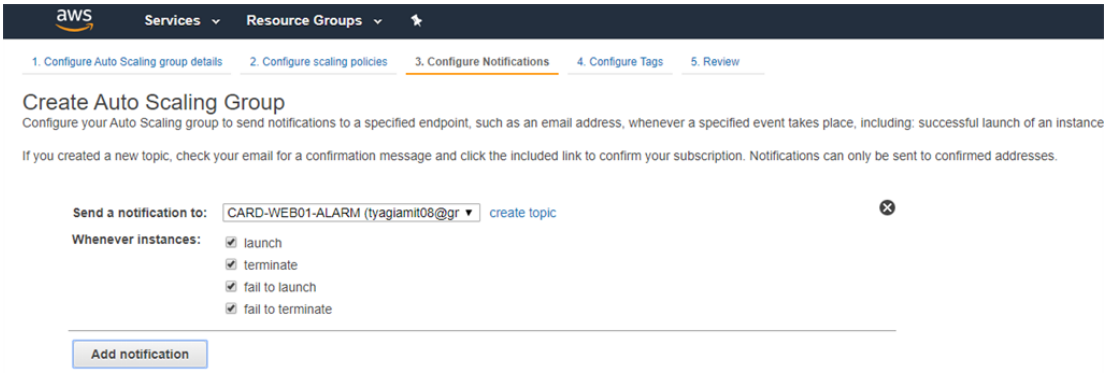



So above it says that if the CPU utilization is > 60 % for consecutive 5 mins then raise this alarm. We will also add alarm for the decreasing group size as if the CPU utilization is < 40 % for consecutive 5 mins then raise this alarm.

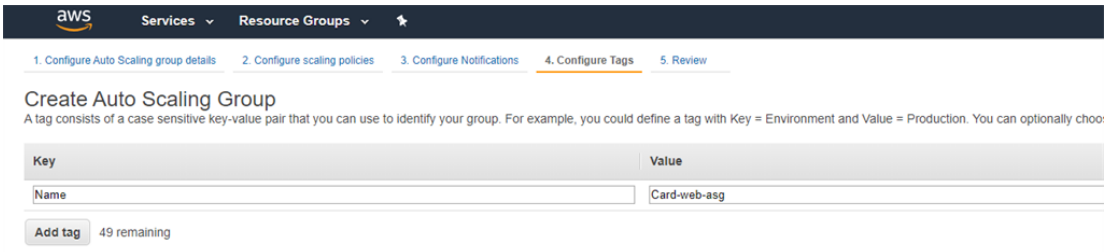




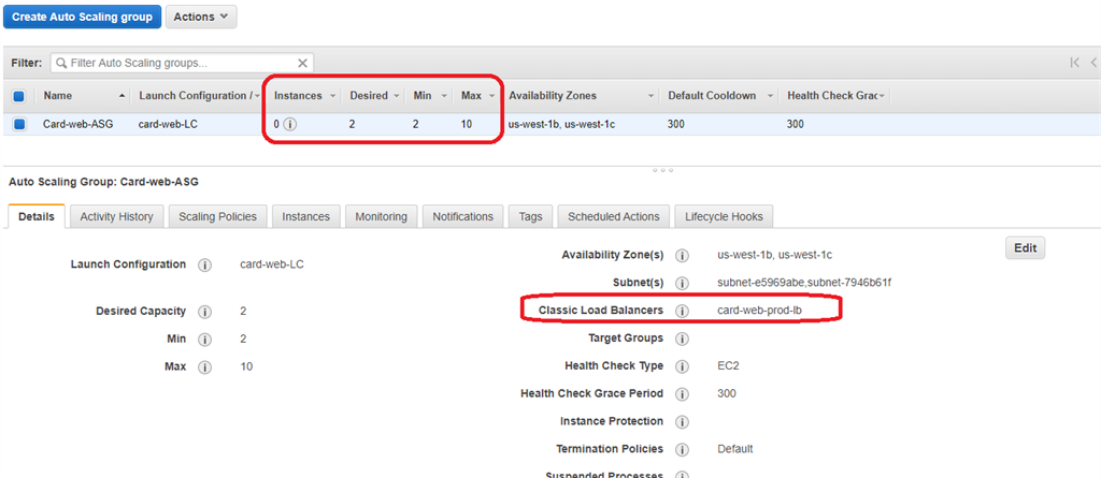

So finally, we can see the complete scaling policies as shown above. Then next is to configure notification, which means when we want the team to be notified.




Then add tags if required.



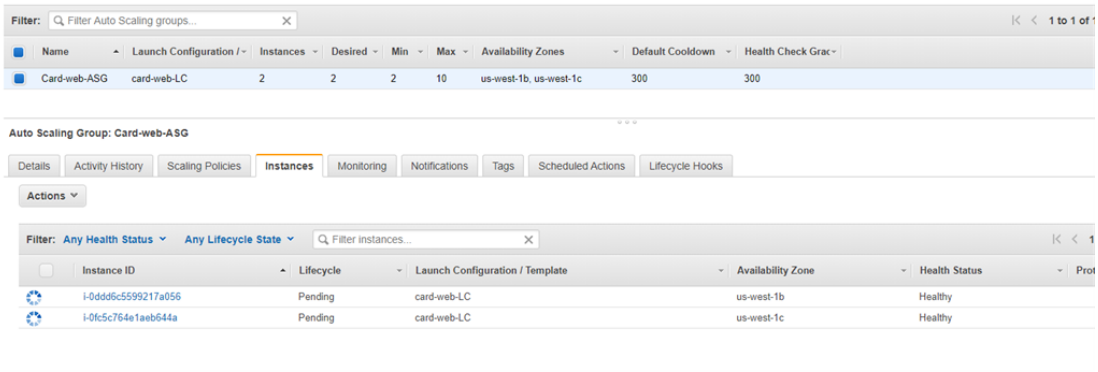

Finally review and click on Create and we can see that our auto scaling group has been create successfully.

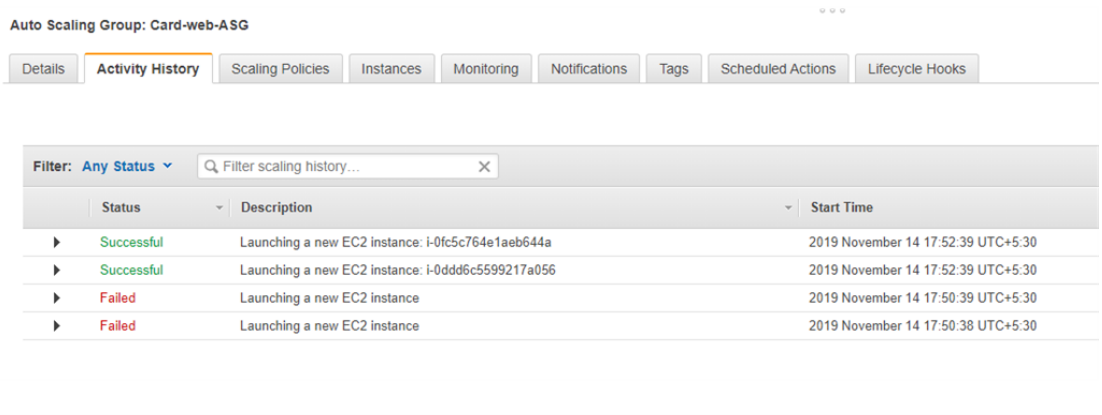


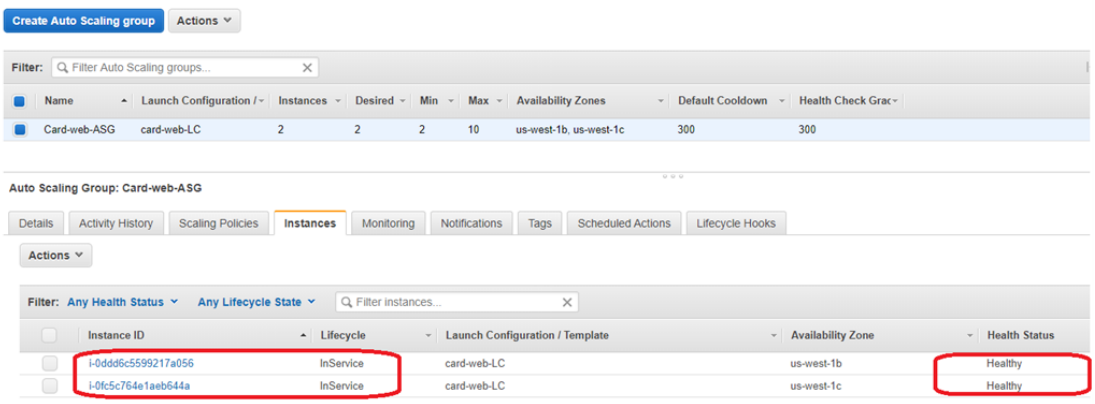

**NOTE**

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.

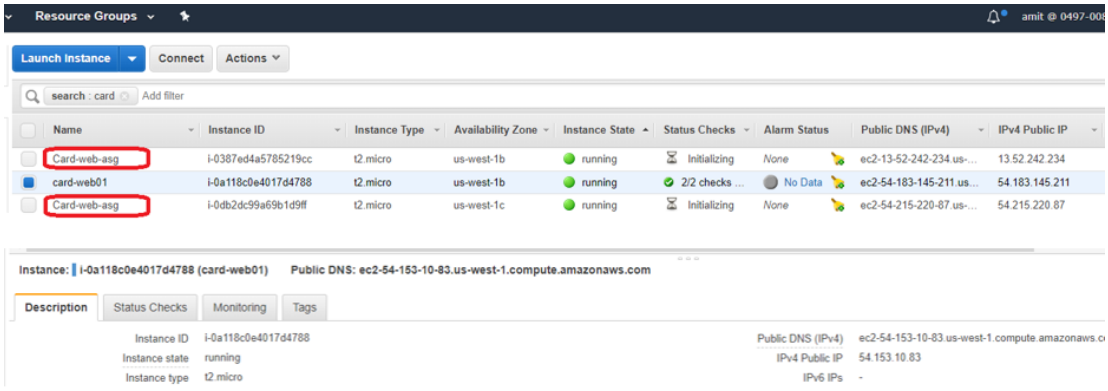
So, first thing it will do is , it will launch the instances and we can see that in the activity history as well.



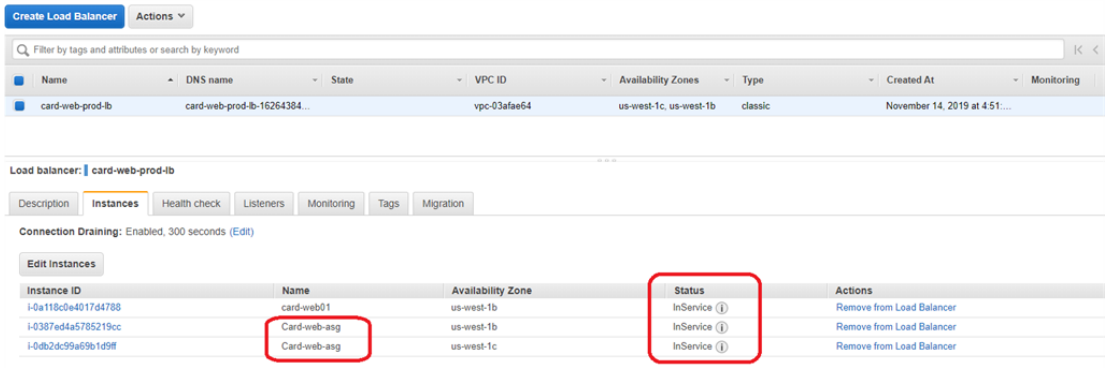
So, the newly launched instances are in service and are healthy and in the instance section we can see the 2 newly launched instances too.




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 .

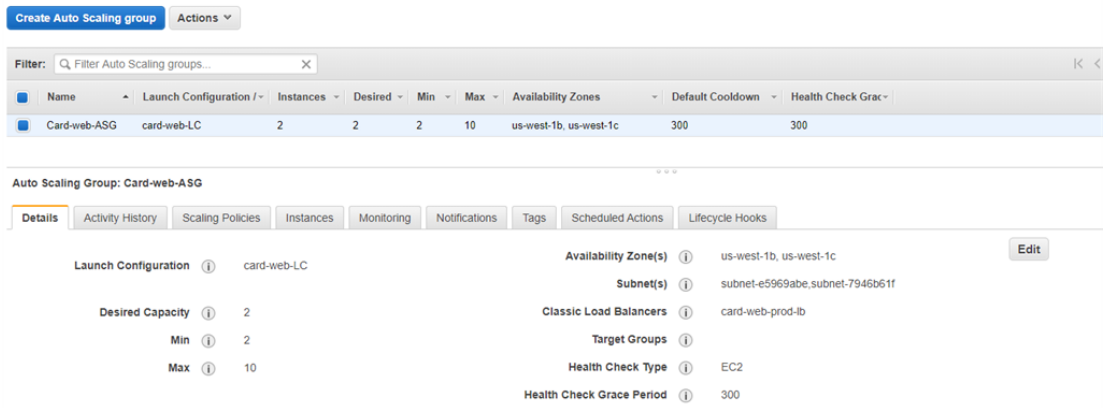
We can see that the 2 newly created instances have also become part of the load balance and are in in service.




# DELETING INSTANCE TO CHECK AUTO SCALING

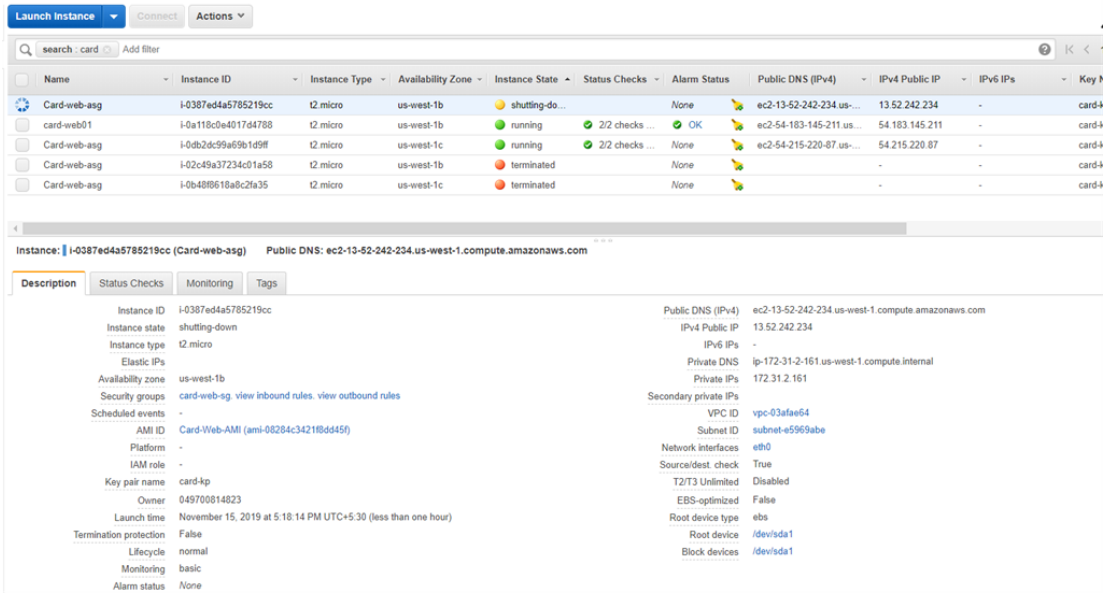
Now as we have done the setup, so let's check if things are working as expected.

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.

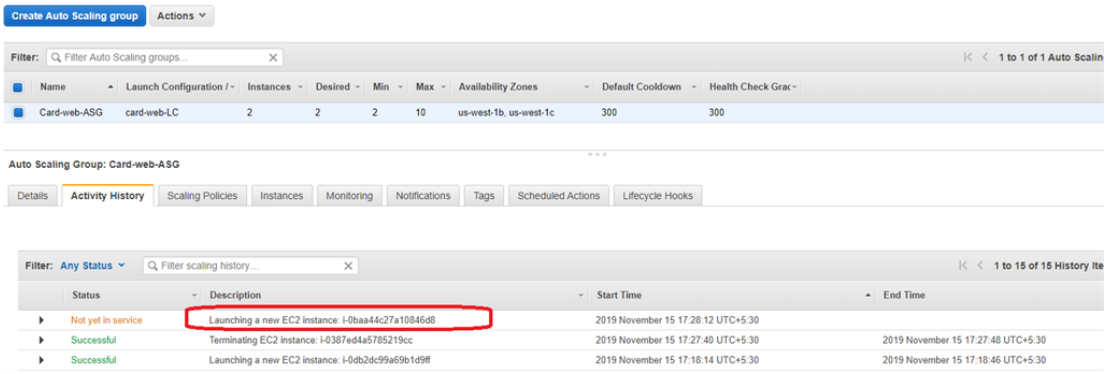



Here we can see that there are 2 instances attached to the auto scaling group. So, if I delete one then auto scaling group will created another automatically.

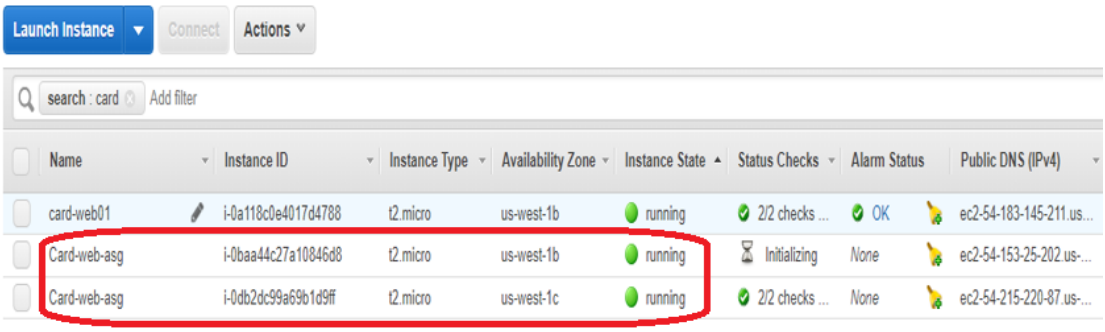
As you can see that we have triggered the termination process on one of the instances.




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




And again, we have 2 instances running.




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!!*