Now the real power of auto scaling groups comes from the **automation**. **So rather than having to change this manually, we can have it so that the auto scaling group automatically scales up and down the number of running instances based on whatever monitoring that we decide on**. Now we can access that by going to the scaling policies tab and right now there are no scaling policies defined.

Now **scaling policies can be based on either something that's monitored so it can increase or decrease the number of running instances so scale in or out based, for example, on CPU utilization but we can also define scheduled actions. So if we know that a website is going to be busy at a certain point in the day or certain periods during the week that we can create scheduled actions and have it automatically adjust the desired capacity based on the load that we expect during those periods.** Generally, though, what you'll find is that most auto scaling groups use scaling the policies, so let's go ahead and add a scaling policy.   
Now **scaling policies come in one of three types we've got simple scaling policies and that allows us to define a rule**. For example, **if the CPU utilization of all of the instances on average is above 30% then add one instance or add two instances. We can also say if it's below a certain amount, then remove one instance or remove three instances.** So this is the simplest type of scaling policy and this is the one that I tend to see more often. Again what we can do for each one of these scaling policies is **define a number of seconds between this action and any successive actions**. This will override that default cool down value that I talked about earlier in this lesson. But if we don't specify anything then it uses that default.   
The second type of scaling policy is the **target scaling policy**, and what we can do with that is we can define a value, say, average CPU utilization and define the value that we want the average to be at. Let's say that we want the average CPU to be around 20%. Well, if we define this, we say 20% it means that if the CPU utilization is above this value, then it will take an action needed to correct that. Maybe it will add instances or remove instances. If it's below that amount, then it will reduce instances. So that's a slight variation on the simple scaling policy, because you're defining an ideal value for a specific metric and allowing the auto scaling group to manage it for you.   
Then we've got **a scaling policy with steps**, and this is slightly different again. This allows us, in much the same way as a simple scaling policy, we can define a certain metric and say if it's above a certain value then take an action, add an instance, add two instances, but with the stepped one we can define different levels. So I don't have an alarm created at the moment but let me select billing alarm just to demonstrate this, I'd be able to take a certain steps so that maybe one or two instances when the value was a certain level to infinity but I could also add another step. Now why you might use this is if you're monitoring the auto scaling group and let's say that CPU utilization goes from 20% to say 50%. You might want to just add one instance but if it goes all the way up to 90%, then you might want to take more serious action may be adding five or 10 instances. So this is generally **used if you're expecting a fairly varied load, and you need to react to it quicker. If you got a simple scaling policy and you decide to add, say, one instance, if the CPU is above a certain amount, then if you do that, remember, you can't take another action until a certain time has elapsed.** So these are three different ways that you can think about scaling. But generally you'd most often use a simple scaling policy, especially for the exam. You're probably going to be tested on simple scaling policy, but I do want you to be aware of what a target tracking policy is and what a scaling policy with steps is.

But I'm going to create a simple scaling policy for this demonstration. Now the name of this policy is going to be CPUHIGH. What I want to do is create a brand new alarm. I don't want a notification and I want this alarm to trigger if the average CPU utilization of all of the EC2 instances inside this auto scaling group is more than or equal to 10%. So every alarm that you define has both a value and then a number of consecutive periods. So I'm going to say that if the CPU utilization is greater than or equal to 10% across this entire auto scaling group for at least one consecutive period, that I want this alarm to trigger. So I'm going to add on the end of this alarm high. So I want this alarm to trigger if the CPU is higher than 10% on average for at least one period of five minutes, so I'm going to go ahead and create the alarm but I also want to do the inverse, so I'm going to add a policy. It's going to be a simple scaling policy. I'm going to call it CPULOW and I'm going to create a new alarm. Again, I'll uncheck notifications and this time I'll say whenever the average of CPU utilization is less than or equal to 10% for at least one consecutive period of five minutes I'm going to name it CPULOW and create the alarm. So this time this is going to alarm if the CPU is less than 10% for one period of 300 seconds, and in that case, I want to remove one instance. So this is the inverse of the previous scaling policy. So I'm going to go ahead and create this policy. It's currently on one instance the desired is set to one, the minimum is set to one, and the maximum is set to three. It's important to understand that the auto scaling group size, as I mentioned at the start of this lesson can never go below the minimum. So even if this CPU alarm is registering as it's below 10% it can never move this auto scaling group to zero instances. That's really important to understand.

Now to test this, even though Winky is the most amazing cat in the world, nobody knows about this website, so it's going to be really hard for me to generate real load on this website, so I'll have to simulate it. So I'm going to do that I'm going to go to instances. I'm going to get the details for the current instance. I'm going to right click and hit connect. Now because I'm running Mac OS, I'll need to change the permissions on the pem key that I'm going to use to connect to this instance. If you're running on Windows, you can use this link to get instructions for your operating system. So I'm going to move across to my terminal and then I'm in my downloads folder and I'm going to change the permissions on this pem key. Once I've done that, I'm going to use this command to connect into this EC2 instance. I'll paste this in because it's the first time I'll need to accept the authenticity but then I'm connected to the instance. So I'll clear the screen, make it a little bit easier to see. So this is the instance that's running the cat website. Now, after I've cleared the screen, I want to install a utility called stress, which allows me to stress test the CPU. So first, I'll need to add the repository that this utility is contained within. So I'll run this command and don't worry, I'll make sure these are in the lesson description if you do want to follow along with this in your own environment. Once I've got the repository installed, I'm going to go ahead and install the stress utility by running this command. Okay, so that's installed and now I want to go ahead and I'll clear the screen first, and then I want to stress the CPU. So I'm going to use this utility so first I'm going to sudo and then stress and double hyphen CPU and here you'll need to specify the number of CPUs that you want to stress this needs to be a value that's equal or greater to the number of CPUs that you have. So I'm just going to specify two at this point and then space and then double hyphen timeout and the number of seconds that you want to run this for. Now, remember that I've got this set up not to use detailed monitoring on CloudWatch and I'll be talking about CloudWatch in the next section of the course. What this means is that monitoring is based on five minute periods, which is 300 seconds. So if I set this to a too low value, it would stop before the monitoring kicked in. So I'm going to pick a value that super high. So 300 is actually five minutes, so I'm just going to take 300 then add a couple of extra zeros. So this is 50 minutes and this is 500 minutes. Once I've done that, I'm going to go ahead and press enter. So this means we're now stressing out the CPU on this EC2 instance, I'm going to move back to the console and what we should be able to see once the monitoring kicks in is that the CPU utilization that's reported rapidly increases. So it's been hovering around the 5% mark but now that we've got this stress test utility running we'll see that rapidly increases. Now, this could take a couple of minutes. So what I'm going to do is I'm going to pause the video and I'll resume it once we start seeing things happen. Okay, so it only took a few minutes, but we can see that the CPU utilization has gone from 6% all the way to above 15 and straightaway we'll see we've got a pending instance. If I go to the auto scaling group configuration and select our auto scaling group, we can see that the desired has increased from one to two and that's based on the scaling policy, which is looking at CPU utilization. So if I go to activity history. This gives an overview of all of the different activities conducted by this auto scaling group. Now the top one, which is launching a new EC2 instance, that's the most recent based on time and if I expand that, I can see the cause for that. So we're launching a new EC2 instance, it gives the ID and then the causes at this particular time a monitor alarm and then we've got the alarm name that I just created. So that's the awssec2 ASG CPU utilization high that is now in an alarm state. So because that's in an alarm state, it triggered the scaling action and that increased the capacity from one to two. Now because I'm still running that stress test, what we're going to see is over time this desired value is going to increase from two to three. It won't happen yet because there is a timeout, remember, default cool down of five minutes between successive scaling actions. So it's going to have to wait five minutes until it adds an additional instance. Now while that's happening, I'm going to go ahead and go to load balancers, and I'm going to create a brand new load balancer. So create load balancer, I'm going to skip through this relatively quickly because I've covered this in previous lessons. So it's going to be an application load balancer. I'm going to call it ALB, internet facing, IP version four, it's going to allow HTTP on the load balancer of port 80. I'm going to use the default VPC. Select the same availability zones is the auto scaling group so 1A, 1B, and 1C. I'll go to next, configure security settings, then next again configure security groups, I'll create a new security group. I'll call it security group ALB. It already allows TCP port 80 which is HTTP and that's exactly what I want. So next configure outing, I'm going to make a new target group. Target group is going to be called cats. It's going to be based on instance ID The protocol is going to be HTTP port 80. The health check HTTP and remember, because I'm using that dynamic IP address from the previous lesson, the health check path needs to be index.php. I'm going to change these health check thresholds to the ones that are used in previous lessons. So healthy threshold is two, unhealthy is two. The time out is going to be four seconds and then the interval is going to be the minimum possible value, which is five. Next I'll go to register targets and we can see that because I've already got this target group used. I created this in a previous lesson. I'm going to pick cats ASG for cats auto scaling group. Next, I'll go to register targets. I'll not manually register any targets at this point because I'm going to integrate this with the auto scaling group. So I'm just going to go to next review and then create the load balancer. So there's nothing different there than what I'm doing previous lessons. Now we'll need to give that some time to finish provisioning. It could take anywhere from a couple of minutes all the way up to 50 minutes, but it will take some time. So in the meantime, I'm just going to open a new tab and go to auto scaling groups. I'll select the auto scaling group I just created, go to actions, and then edit and what I'm going to do is associate this auto scaling group with the load balancer. Now, if I was using a classic load balancer, what I'd be doing would be associating the auto scaling group with the load balancer directly. But because this is an application load balancer, what I'm actually doing is associating it with a target group, something to select target groups and go to cats ASG which is the target group associated with the load balancer that was created. So I'm going to select that and what that means is that any instances created by this auto scaling group will automatically be registered with this target group. And any instances which are terminated by this auto scaling group will be removed from this target group. Now, I'm also able to specify the type of health check that I want to use, remember, an auto scaling group is going to automatically terminate any instances which are deemed to be unhealthy. By default, that health check is based on the EC2 health check type, which essentially uses the status check of the instance itself, as well as the instance host. Now, because I've integrated it with load balancer, I can make use of the load balancer health checks. So if I select ELB that changes the health check type to the load balancer health check and that means that any instances which fail the load balancer health check will be terminated. Now, what that means from the auto scaling group is that because a terminated instance reduces the running instance count, that will mean that the running instances is below the desired capacity and that will mean that it will create a new instance in its place. So by adding an auto scaling group to a load balancer and changing the health check type to ELB you actually allow your application platform to auto heal, so it'll automatically heal any failed instances. So that's done. I'm going to click on "Save." So I'm going to go back to the list of EC2 instances, and I'm going to pick the one that was launched most recent. So that's the one that's been launched at 12:16. So I'm going to select that one because that will be the last one that gets terminated. I'm going to connect back to this instance. I'll acknowledge the authenticity, and as soon as I'm connected, I'm going to go ahead and stop the web server. So sudo service httpd and then stop. So that's going to stop the web server running on this EC2 instance. What that means is it's going to fail the load balancer health check. So if you look at load balancers, go to listeners, go to the cat auto scaling group, target group we'll see that it's now in an unhealthy state and once it's gone past the two consecutive periods, so it needs to register to fail checks to go to an unhealthy state. What we'll see is because the auto scaling group is now using this as a health check we'll see after a brief amount of time that it will terminate this instance and once it's terminated, it will recreate a brand new instance in its place. So I hope you can say at this stage that by using a load **balance it together with an auto scaling group and the launch template, you can implement a highly available, elastically scaling, and self healing piece of infrastructure, and you can use that to host your applications, and you don't have to worry about any of the usual high availability or scaling concerns that you have with non AWS infrastructure**. Now that's everything I wanted to cover. I know it's been a lot of content in a big grouping of lessons, but it's really important subjects that you need to understand both for using this in production and for the SA associate exam. So go ahead, mark this lesson as complete and when you're ready, you can join me in the next topic of the course where we're going to be covering VPN and Direct Connect.