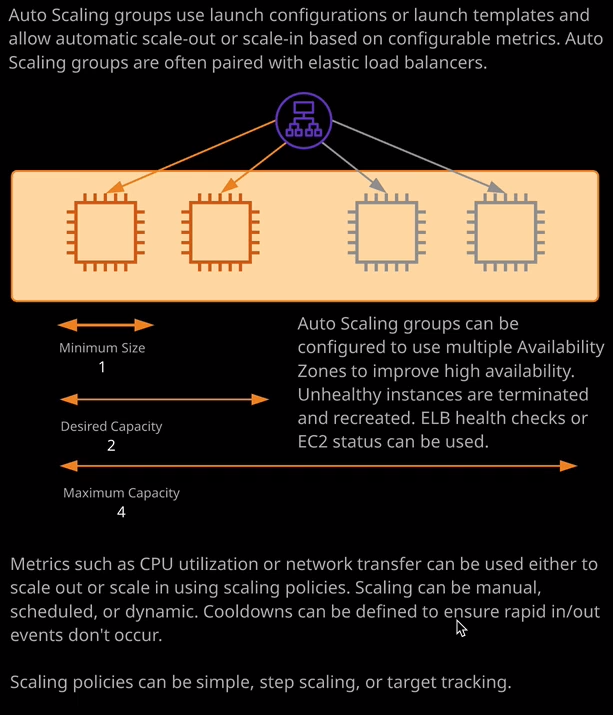
Welcome back and in the last lesson, I talked about launch templates and launch configurations which are ways you can define what you want to launch and what configuration you want EC2 instances to use. In this lesson, I'm going to be talking about auto scaling groups which is the second component of a highly scalable architecture inside AWS.

So **auto scaling groups utilize launch configurations or launch templates, so they're used to define the what and then auto scaling groups define exactly how you want those instances to perform**. So exactly how you want them to scale. Under what circumstances do you want your instances or your software to scale out by adding instances or scaling in by removing instances? Now, earlier in the course, right at the very beginning, I talked about the different types of scaling architecture, and I introduced a term known as elasticity.

**Now elasticity is a process that's been popularized by AWS**. **It's essentially a process where you can define a platform or set of infrastructure, which scales out when load increases and scales in when load decreases.** But what's special **about elasticity is that it's generally fully automated and happens in real time based on the load on that platform, so you don't have to manually increase the number of EC2 instances that are available for your web application. Auto scaling and elasticity handle this for you so they grow your platform as the load on those servers increases and they reduce it as load decreases and that means that you're always paying an appropriate cost for infrastructure deployed into AWS**. Now, when you combine this with a load balancer, you make it so that any of your customers that are abstracted away from the exact number of instances that you have. So your customers talk to a load balancer and then the load balancer automatically integrates with the auto scaling group and distributes incoming connections across that auto scaling group. So that's something that I want to cover as well in this lesson. So I'm going to talk about auto scaling groups and then how we can integrate them with load balancers.



Okay, so to start, I'm going to go to auto scaling groups. I'm going to click "Create Auto Scaling Group" and if you do in this from an environment where you don't have either launch configurations or launch templates defined than it will show you a wizard where you'll be asked to create one but I wanted to illustrate this as a separate protest because this is how you're going to do it in a production setting. So I'm going to select to use a launch template because this from now on, as I talked about in the previous lesson, should be the preference. So I'm going to go with launch template and I'm going to select the launch template that I created in the previous lesson. So I select that and hit "Next Step." Now I'll need to name the auto scaling group. So to keep things simple, I'm going to call it ASG. Now, because I'm using launch templates I will have the option of picking a version. So you have a number of different options. You can pick the default version so whatever set to be the default version of this template, you can select latest which could be the same as default but it could be different, or you can explicitly pick a particular version. Now, in this case, the latest, the default, and version one are all the same thing. So that's why you can see default listed here next to one. So it doesn't matter in this case. So I'm just going to go ahead on pick version one explicitly. Now for fleet composition, remember, you are able to specify the purchase option inside the launch template so you can specify to use spot instances. At this point, you can either use whatever the launch template says, or you can combine purchase options. Now because I'm not using any spot instances. I didn't specify that configuration. I'm just going to leave this as default and in most cases, unless you've got some particular purchase option demands, you can leave this as default. Now, at this point, I want to stress a couple of really important things to realize about auto scaling groups. I'm just going to expand this advanced details just so we can see all of the options.

Now when it comes to auto scaling groups, there are three main configuration values that you need to be aware of. There's the minimum size of the auto scaling group, and that is the absolute minimum number of instances that the auto scaling group can have in it. So if it's set to one, even if there's no load on the platform, it will, at minimum, always have one EC2 instance, then we also have maximum capacity, and this is the maximum number of instances that the group will ever grow to. So if we set the maximum to be four, then the capacity can range between one and four but even if every instance is completely overloaded, it will never grow beyond the maximum capacity. So you can think of this as a cost control value. You don't want to set it too low because it can impact the performance of your application, but you don't want to set it too high because that could massively increase costs. Now the desired capacity is the number that the auto scaling group will attempt to aim for. So if you've currently got one EC2 instance and the desired capacity is set to two, then the auto scaling group will attempt to create a new EC2 instance to bring the number of running instances to the desired capacity. If you set it to three it will add an additional instance. If the desired capacity is set lower than the number of instances, then it will terminate instances to bring down the running number to whatever the desired capacity is. So at a high level, the way that an auto scaling group works is it uses certain monitoring metrics, and based on that it increases or decreases the desired capacity and then the auto scaling group either terminates instances or creates instances in order to match that capacity and it creates instances using the configuration of whatever is set in the launch template or the launch configuration. So that's, at a really high level, exactly how it works. So what you need to configure from an auto scaling group perspective is you configure the number of instances to start with, so I'm going to leave this as one. You need to specify a network to use. So in this case, it's the default VPC, and you need to specify which subnets to use for this auto scaling group. Now I can go ahead and select all of the available subnets, which I'll do so 1A, 1B, 1C, D, E, and then F. Now I'm creating this in North Virginia, so that means I've got six availabilities zones. This may differ depending on what region you're located in, but this is defining which availabilities zones this auto scaling group can use. So if I specify all six then it will use all six. If I reduce this down so maybe I cut out D, E, and F and this auto scaling group will only use 1A, 1B, and 1C. Now it will attempt to equalize the number of instances that are running inside each of these availability zones. So if I've got two instances that are currently running one that's in 1A and one that's in 1B and I launch another instance, then generally it will put it in 1C, so it tries to even the instances across the available subnets. Now, this is where auto scaling groups help you with high availability because once we pair this with load balancer, it tries to make sure that all of our compute, so all of our instances, are distributed across availability zones and that increases our resilience to the failure of any individual availability zone. Now, I'm going to configure load balancer integration as a next step, so I'm going to leave this is default for now, but what I want to talk about is a health check grace period. So when an EC2 instance is provisioned inside an auto scaling group, logically, it will take some amount of time to perform whatever auto configuration you've got set inside the user data and it could be relatively quick in the example that I've been using of building a simple web server but if you're installing anything more complex, it could potentially take longer. So you always need to make sure this health check grace period is set to whatever time your configuration requires plus a buffer. By default, the health check will be an EC2 instance status check. So it's mainly checking that the instance itself is running and the physical hardware that the instance is running on is having no problems. But as you'll see when you integrate load balancing, you get the option to use the load balancer health check, which adds additional capability, as you've seen in earlier lessons. So I'll leave the rest of this as default. I'll go to configure scaling policies and for now I'll leave this, but keep the group at its initial size. I'll talk more about this in a second. So I'll go to configure notifications. You are able to add notification settings. So if any scaling occurs, you can have it send a notification out to, say an SNS topic but again, I won't want to do that for this demonstration. So I'm going to go to next configure tags. You can configure tags on this auto scaling group and have them populate out to any instances that are created by it. So you could, for example, have ASG as the key name and then the value you could have set to whatever the name of the auto scaling group is. So any instances that are created you'd be able to easily relate them back to the auto scaling group that created them. I won't be doing this today so I'm just going to go ahead and delete the tag and click "Review" and then create auto scaling group now, because I haven't set any scaling policies because I set the initial size to one that's going to set the desired to one, the minimum to one, and the maximum to one. Now, remember when I talked about the desired capacity, I said that the auto scaling group is going to attempt to change the number of running instances to whatever the desired is set to. So right now, the number of running instances is zero and the desired is one. So that means it's going to attempt to create an EC2 instance, if I go to the instances console, you'll see that it's already created an instance, and it's currently initializing and because I've used the launch template inside the auto scaling group, this instance is going to be automatically created and configured running the cat web website. So if I select this instance and I copy of the DNS name into my clipboard and then open that in a new tab, you'll see that it's running the website that's showing the current public IP address on the picture of Winky, my cat. So let's go back to the auto scaling group configuration and if I go to actions and then edit the auto scaling group, I'm going to change the maximum to three. So we've currently got it running inside three availability zones so I'll change maximum to three and then at the same time, I'm going to change the desired capacity to two and click on "Save." What that means is the desired capacity is now larger than the number of running instances. So that's going to go ahead and create a brand new EC2 instance.

I want to draw your attention, though, to this **default cool down setting. What default cool down does is it puts almost like a pause timer between scaling events**. So this is in seconds and what this means is that it's a scaling event, and I'll talk about what they are in a second, **if a scaling event happens for an auto scaling group, let's say that it adds an EC2 instance, default cool down is going to mean that no further scaling events can occur for 300 seconds and this avoids the situation where you can have really spiky load and have instances being added and removed in rapid succession.** So this is something that can come up in the exam because, of course, **if you terminate an instance and recreate it and then terminate it and recreate it, there is a minimum billing period for an EC2 instance. So by rapidly terminating and recreating, you can end up in a situation, especially if this is automated, where you're experiencing significant costs. So the default cool down time allows you to specify a pause in between successive scaling events.** So that's important to understand that something that does very regularly come up in the exam.

So if I go back to instances and just look exactly what's occurred because I've changed this from the default—so I've gone from desired of one to desired of two because there were no scaling events that happened before—then it's gone ahead and created a second EC2 instance and once this is finished creating again, this will be running the same cat website that the first server is running because it's using the automated configuration as part of the launch template. I'm just going to go back to auto scaling groups for now. What I'll do is I'll go to actions and edit, and I'm just going to drop the desired back down to one, and then we can move on to the next part of this demonstration. Because the desired is now less than the number of running instances, what we'll find is that one of these is going to be terminated, and we can already see that occurring now. So this instance is being shut down and it's being shut down because the auto scaling group sees that it's got more running instances than desired.