

Welcome back and welcome to this lesson where I'm going to be talking about Aurora Serverless. So far on this topic of the course I have been discussing Aurora provisioned, which is essentially a version of Aurora that's provisioned inside your AWS account in a similar way to previously available database engines inside RDS. **Aurora serverless is AWS's answer to being able to provide an on-demand auto scaling configuration for Amazon Aurora delivered as a service. So if you want a relational database, but you want it delivered as a service with no admin overhead whatsoever, then one option is to use Aurora Serverless, and it's a really good database product because it offers the ability to scale in capacity. Aurora serverless uses a concept known as Aurora capacity units or ACUs and these are essentially an abstraction away from physical hardware specifications. You essentially specify for a given Aurora serverless cluster a minimum ACU and a maximum ACU and transparent to you, AWS handles the scaling without any disruption to an application, and it does that by using a proxy fleet. A proxy fleet is essentially a transparent set of proxying instances that grow and shrink based on demand and the function of this proxy fleet is to essentially transparently direct connections from your applications to these instances that are running inside Aurora serverless without your application knowing any different. So essentially, you simply create this cluster, specify a minimum and maximums, and everything else is handled by AWS.**

Now, the reason **why you would use Aurora serverless is it removes much of the complexity of managing database instances and capacity. So one of the things about RDS as a product is that it's an abstraction away from self-managed database products. So to use RDS, you provision a database, and you can access it just as you would do if you were managing that database product.** But as I've demonstrated in earlier topics of this course, **by using RDS, you remove the requirement to provision your own hardware or your own virtual machines, install the database software, manage backups, manage high availability, and manage performance. A lot of that is removed by using RDS. Aurora Serverless is essentially just one more step down that process, where, when you're using Aurora Serverless, you don't even have to worry about the server instances themselves. It's all handled end to end. When you use Aurora Serverless, you simply pay for the database resources that you use on a per second basis and if you exceed the capacity of what you got access to, transparently, behind the scenes, Aurora Serverless provisions a new, larger database instance or multiple smaller database instances, transparently adds them to your cluster, and then removes the small ones which are no longer being used. Essentially, it is a truly as a service database product,** so let's have a look at exactly how it works.

So if I go to databases this is the Aurora provisioned cluster that I have been using in previous lessons and at this point in time. It's got a writer instance and a reader instance. Now whether I'm using the capacity inside this cluster or not, it is still costing me money to run both of these roles and I can use a feature, which is **auto scaling, which is available inside Aurora provisioned to dynamically scale up and down the number of reader instances**, so that's important to know for the exam. But still there's always going to be a minimum and the level to which I can increase or decrease these reader instances is always going to be limited. We're not going to get that nice linear alignment between the capacity that we need and the capacity that is provided when we're using Aurora provisioned. So Aurora Serverless attempts to provide a better solution for that. So if I go to create database and then I select Aurora and I choose Amazon Aurora with mySQL compatibility and if I scroll down, I'm able to select Aurora Serverless when I select Aurora Serverless a lot of the options that we've seen in the previous Aurora provisioned lessons disappear. So I'm still expected to provide a database cluster identifier, a username to log in to the Aurora Serverless cluster, so that's username and password, but this time we're asked to provide a minimum and maximum capacity. Now this uses the Aurora capacity unit, and I'll make sure that I include a link in the lesson description, which details exactly what this means. Essentially, it is a unit of compute or an abstracted unit of compute and memory, so you can specify a minimum capacity unit and one capacity unit, for example, provides 2 gig or two gb of RAM. You can also specify a maximum. So in this case, four capacity units is 8 gb of RAM. So let's say I selected that then this Aurora serverless cluster could scale from one to four as load requirements dictate. Now under additional options, I've also got the ability to pause the compute capacity after a certain number of minutes of inactivity. So unlike Aurora provision, you do have the ability to scale right down to zero instances in this server cluster. Now, just like Aurora provisioned, Aurora Serverless uses this shared cluster storage. This always exists even if you scale down to zero capacity units you still maintain your shared cluster storage, and it operates in exactly the same way as Aurora provisions. **So you're only billed for the capacity that you consume. Aurora Serverless still operates inside a VPC. So it is not a true public space product just like S3 or DynamoDB** that we're going to talk about in the next topic. Now, **as well as being able to provision an Aurora Serverless cluster from scratch, you can also restore it from a snapshot**. If I go to snapshots, I've got a snapshot that I actually took before I started this lesson of the Aurora lesson two database instance and that's currently storing the data with our cat pictures blog complete with the Dutch cat and Dutch flag. So what I'm going to do is restore this snapshot to an Aurora serverless cluster. So I'm going to select it, click on "Actions," and then restore snapshot. Now, if I was restoring this to a provisioned cluster, then I need to set all of these options that by now you'll be familiar with. But what I'm going to do instead is select the serverless option. When I select that the options change. I still need to specify a database cluster identifier and I'm going to do that but I'm going to call it Aurora Lesson Serverless. I'll scroll down and select one Aurora capacity unit minimum and for the maximum, I'm going to change this to two capacity units. Now, I'm going to expand this scaling configuration and allow the cluster to be paused after five minutes of inactivity. So I'm going to set that option. I'm going to scroll down, make sure it's set to use VPC one and the Aurora subnet group and we're going to choose an existing security group, going to drop down and change this to Aurora Security Group. So this is going to mean that I don't have to make any modifications to our WordPress instance apart from changing the DNS name and the login information. So this is the security group that I configured in previous lesson when I set up the Aurora provisioned cluster. Now our Bastion host is running with a security group attached to it, which has access into this security group. So by choosing this, it just means I've got one less step to configure. Everything else looks good. I can leave this as default and go ahead and restore a database cluster. Now, this is going to take a couple of minutes to fully provision and while it does that, I'm going to select the Aurora Serverless cluster. What I want to do is copy the endpoint address. So this is the DNS name that references the cluster. So I'm going to copy that into my clipboard and move back across to my terminal and I'm going to update the wordpress-config file with this endpoint address. So rather than pointing at the Aurora provisioned cluster, I'm going to have it pointing at the serverless cluster. So I'm going to delete this DNS address that exists in the file already. So I'll just delete this and then I'll paste in the serverless cluster. I'll save the file and then exit out of this configuration and that's all the change that we need to do because we're restoring this from the snapshot. The database is going to be the same and the username and password are going to be the same, and it's still in a creating state. So this is going to take a couple of minutes to finish up. So I'm going to go ahead and pause the video and resume it once this is ready to go.

Okay, so at this point, this cluster is finished provisioning and we're currently running at two capacity units. So if I go to the cat pictures blog and just to refresh, just by provisioning the cluster from the snapshot and changing that endpoint DNS name in the WordPress configuration file, we're now using Aurora Serverless. **So this is now a database instance which doesn't have any specific instances allocated to me, it's only using capacity units. Now it's currently using two capacity units because I've provisioned it. What will happen is over time as we're not using this cat pictures blog, it's going to scale back these capacity units even going to zero**. So if there's no load on this blog whatsoever because we've specified that we want to be able to pause it, it will scale back to zero capacity units. Now what I'm going to do is I'm going to close this tab and just let it do just that. So I'm going to try to get it to the point where it will pause. So if we hope for a second nobody is scraping the website, no search engines or any other activity, then after about five minutes, it is going to scale back and pause the cluster and I want to demonstrate exactly how that works.

Now while that is happening, I want to talk about the **ideal use cases for Aurora Serverless** because I think it's important for the exam that **you can distinguish between when you'd use Aurora Serverless and when you'd use Aurora Provisioned. So I want you to imagine a couple of scenarios. Maybe you're deploying a new application and you're unsure about its usage patterns. You could just provisions that smallest Aurora provisioned cluster that you can. But if you do that, then you're subject to these uncertain load patterns. So if you do get heavy load really early on, then you're going to get some performance issues. Now one of the scenario is that you've got an application that receives fairly light use. Maybe it receives 2 or 3% of the maximum load during a normal 24/7 period. But there are periods during the day when it peaks, and when it does peak, maybe you get 30 or 45 minutes of really high usage. If you were provisioning an Aurora provisioned cluster to make that happen, you would need to have a single writer instance and then scale out the number of reader instances. With Aurora Serverless, we could define the minimum and maximum ACU values and allow the cluster to scale out as required. Another use case is that maybe a developer is using databases during work hours, maybe for Dev, for test, for staging, or for A/B testing, any sort of development activity. Well if you use Aurora provisioned, you've got to have that infrastructure open running 24/7. You could shut it down, but that's an explicit manual action. With Aurora Serverless, you could set the cluster up in much the same way that I've done here, so you've got the minimum and maximum values, but you allow the cluster to scale down to zero capacity and pause when it's not required.**

**With Aurora Serverless, you've got all of that flexibility. You can treat an Aurora Serverless cluster, just like you would an Auto Scaling Group or any other scalable AWS product. You define a minimum, a maximum, and you allow it to scale as demand dictates.** So let's have a look at how this is going. Let's just refresh this serverless cluster and it still is running with two capacity unit, so it still does look as though it is active and doing something. Let's have a look at the monitoring tab and it looks as though the CPU utilization is dropping. So we are getting towards the low levels what we want. So I'm expecting the capacity of this to reduce. So what I might do is pause the video and I'll resume it once I notice a capacity change. Now this lesson is getting a little bit on the long side. So I wanted to split it into a number of different parts and give you the opportunity to take a small break. So this is the end of part one. Go ahead and mark this lesson as complete and when you're ready, you can join me in part two.