Welcome back. This is part two of this video we're going to carry on immediately from the end of part one. So let's get started. Okay, so that process took about 15 minutes in this case, it can take longer if you're deploying a multi availability zone instance of RDS but in this case, because it's only operating in a single availability zone, which is U.S. East 1a the process was relatively quick, but at this point I have got an operational RDS instance that I'm ready to use. So let's just open up this RDS instance and have a look at some of the options and information that we've got available. Now the way that you connect to a database inside RDS is that every database instance has an endpoint. It's a cname that points at whatever the current primary instance of this database is. So in this case, I could use this endpoint and this port to connect to this database as though it was a mySQL database that I was managing on my own infrastructure. As long as I had connectivity through the security group I've allocated, then I'd be able to connect in and interact with this database and towards the end of this lesson, we're going to do just that. Now this cname will point at whatever the primary instance is. In an upcoming lesson when I talk about how you can define a primary and a standby instance than this cname could move between the primary on the standby enduring fail over events. So that's how RDS achieves a really high level of resilience. Essentially, you always connect using the cname. You can only ever talk to the primary database instance. You don't have access to any of the other instances directly using this cname it only ever points at the primary. Now I can click on each of these tabs and just demonstrate that we do have a full set of monitoring via CloudWatch we're able to access logs and events via CloudWatch logs and obviously if I'd have configured it to push those into CloudWatch logs I'd be able to access them from the CloudWatch logs console. Now RDS does perform a full set of maintenance and backups. We are talking about backups in an upcoming lesson but it is performing automatic maintenance as per the maintenance window that you defined when you created the instance. Now you're able to perform a number of different actions on database instances. You can stop the database instance, you can reboot, you can delete it, you can take a snapshot, restore to a point in time, and we'll talk about those in a lot in the next lesson when I talk about backups and restores, you can create a read replica and I'll talk about read replicas in an upcoming lesson. So you've got a full set of actions that you can perform on these instances, you're able to modify the database instance so you can change its size. You can add multi AZ capability by adding a standby node, you can change this storage type from general purpose to provisioned IOPS. You can increase the allocated storage size or enable storage auto scaling. You can specify a new master password if you didn't note it down during the creation process and, of course, you got the ability to change the security group. You can change its public accessibility options, configure database ports, parameter groups, enable IAM authentication, enabled backup, or other elements. You got the ability to adjust most of the configuration elements that you used when you created the database instance. For now, though, I won't be changing any of this so I'll just click on "Cancel."

Now as with EC2 you can make use of reserved instances in order to reduce the cost of an RDS instance. With RDS you've got different pricing components. You've got the instances and instance sizes that are used, as well as the fact that you're using an on demand billing model. Now I'll make sure to include a link in the lesson description, which talks about RDS pricing. But if you do have any long term needs for an RDS instance, you can purchase a reserved database instance and that will basically exchange some of the flexibility so you won't be able to achieve savings by powering these things off if you use reserved database instances, but you will get a reduced cost by committing to this term upfront. So this works in much the same way that EC2 reserved instances work so you can purchase the capacity upfront in exchange for a cheaper ongoing cost. So with RDS pricing in general, you've got the cost of the instances, you've got the cost for the storage that's provisioned, and it's important to realize that it's provisioned, not consumed. So it's the amount that you allocate for the database instance. In my case, if you remember, I allocated 20 gig for this instance, and so I'm immediately billed for 20 gb not the amount of storage which is consumed, which at this point is close to zero. There could be extra cost, such as provisioned IOPS. So I'm using general purpose SSD storage and if I wanted to use provisioned IOPS then I'll have the extra associated cost, no data transfer also has a cost. Transferring data out of RDS has a fee, but much like many other AWS services transferring data in is free of charge. With backups, which we'll talk about it in an upcoming lesson. You get 100% of the space that's provisioned for the database included, so that means that you can provision backups and have them operating up to the amount of storage that you provision included in the price, which means in a lot of cases you have very little, if any, ongoing backup costs. Now RDS does have some limits which aren't directly relevant to the exam and I'll make sure that I include a link in the lesson description which talk you through some of these. I won't be bothering talking about there now because it's not super relevant for the exam. At this point, what I want to do is demonstrate how we can connect to this RDS instance. So I'm going to do that now. Now, to show you how this works I'm going to click on "Services" so I'm going to move to the EC2 console, go to running instances, and I'm going to connect to this bastion host that was created by the CloudFormation template. So in this particular case, it's functioning is a bastion host and a web server. So I'm going to right click hit "Connect." Now I'm using Mac OS, which is a Linux or Unix compatible operating system. So I'll be using a set of instructions that won't work if you're using Windows. If you're using Windows, you need to follow the instructions that are contained in this link and I'll make sure this is linked in the lesson description, because I'm using Mac OS however, I'll need to move to my terminal and change the permissions on this pem key to connect to this Bastion host and then I'll want to copy the connection string into my clipboard and paste in and connect to the EC2 instance because it's the first time I'll verify the authenticity that will put me at the shell and then I'll clear the screen to make it a little bit easier to see. Now to demonstrate using this RDS instance, I'm going to quickly perform an installation of WordPress on this EC2 instance. Now to do that, there are a number of steps that I need to do first. So first I'm going to sudo yum update -y and this is going to do a general system update on this EC2 instance. It is going to make sure all of the packages are up to date. Now, this will take a few minutes to finish but once it's done, I'm going to go ahead and install some additional packages. So I'm going to run Amazon amazon-linux-extras install -y and I'm going to install lamp-mariadb10.2-php7.2 and then also the PHP 7.2 package itself. So this will install PHP 7.2 as well as the libraries required by PHP to connect using MariaDB or using mySQL to this RDS instance. I'm going to go ahead and install both of those. Once that's done, I'm going to do one more command. So sudo yum install -y and then httpd. So I'll just make sure that I've got a web server configured on this EC2 instance. Once I've done that, I'm going to make sure that the web server is started and configured to start up every time the EC2 instance boots to do that or run a sudo systemctl start httpd and then immediately following that sudo systemctl enable httpd and both of those together. Just make sure that the web server is started and starts up on every reboot of this EC2 instance at this point to verify that that works, I'm going to switch back to my console. I'm going to get the DNS name of this EC2 instance and open it in a new tab and if everything's working, I should get this test Apache page, and that means I'm good to go. So now I'll move back to the EC2 instance and I'll download WordPress so I'll run a wget and then there's always this URL that will always get you the latest version of WordPress. So it's https://wordpress.org/latest.tar.gz and that will always download the most recent version of WordPress. To extract that I'll run tar -xzf latest.tar.gz. So now I've got that WordPress folder, I want to copy that to my web root folder so that I can use it with the web server. So run a copy -r for recursive. I want the WordPress folder and all files inside that folder, and then I want to copy that to /var/www.html I might get some permissions errors, and if I do, I'll need to run sudo to make sure I can copy that without any issues. So I'll do that to make sure copy those files without any issues. So now I'm going to move to that folder. So cd/var/www/html. I'll clear the screen to make it a little bit easier to see, and then I'll do an ls -la to do a listing of all the files that are in this folder. So the next step is to configure WordPress to point at our RDS instance. So I'm going to go ahead and do that so I'll run a sudo copy and then there's already a templated configuration file that's present in this directory and that's one that's called wp-config-sample.php. So I'm going to go ahead and copy that and I'm going to name it wp-config.php which is the file that WordPress expects. So I'll run that command, and then I'll follow that up by editing this configuration file. Remember, what I'm trying to achieve here is make WordPress point at our RDS instance. So sudo nano wp-config.php and what I need to do is change these definitions to make it so that it points at our RDS instance. So that's what I'm going to do. Step one is to provide the database name and if you recall when I created my RDS instance I called it rdslessondb Next you do need provide the master username recall that was admin. You could set this when you create an RDS instance but I left it as default so it was admin. Next, we need to give it the mySQL RDS password. Now, this was the one that was auto generated by RDS. So I need to get that from the text file where I noted it down when I first created this RDS instance. So I've got that and I'll paste that into this box. Now lastly, we need to specify where the database is. We need to specify the database host name, so that's the next step. To get that, I'll go back to the console. I'll open up the RDS console and it's this database instance Cname that I'll need. This is the name that references the primary database instance. So this is the one that I'll need. So I copy that into my clipboard, move back to my terminal, and paste that in. Now that's everything I'll need to change. So I save that file out and we're almost good to go. There's one last step that we need. Now, when I created this RDS instance, if you recall, I created a security group that this RDS instance uses for security purposes. So I need to edit this security group. At the moment, the Bastion host has no method of connecting in to this RDS instance. The only IP address that's allowed in is my IP address. So I need to edit this RDS security group. So what I'm going to do is I'm going to add another rule. It's going to be mySQL/aurora. So it's TCP ports 33.06 but in this particular case, I'm going to allow connectivity from the Bastion Security Group. So this is a security group that the Bastion instance is in. If you recall from earlier lessons in the course, I talked about how security groups could reference each other, and this is taking advantage of that functionality. So I'm allowing any EC2 instance that has this Bastion security group associated with it to connect in to this RDS instance. So I'm going to make that change and click "Save" and at this point, if I go to the EC2 dashboard, I've configured it so that this instance can now run PHP. So I've installed PHP on the instance, I've installed a web server, I have configured it so that I can connect to this instance. I can browse to the WordPress installation script, install WordPress. WordPress will be able to connect to the RDS instance because of that configuration file that I've edited. But at no point have I had to actually install or manage a database server, and that's the power of RDS. Now this should already work but there's one final set of tweaks that I want to do just to make sure that we don't have any problems with the WordPress instance. So I need to address some permissions. I'm going to run sudo and then chown -R apache /var/www I'm just making sure that the ownership on this folder is correct. I'm going to do the same process for sudo change group -R Apache and then /varr/www then I'm going to adjust the permissions. So sudo chmod 755 again. Same folder /var/www and then last l am going to run two commands and don't worry they're well beyond the scope of what you need for the exam. This is just specifically to set a WordPress. Lastly, I'm going to run a sudo find for /var/www -type d -exec sudo chmod 2775 {} \; and then a semi colon and the same command again but this time changing the type from D to F and the permissions from 2775 to 0664. Again, you don't need to worry about why I'm doing this. This is specific for setting up WordPress but I want to make sure that I don't have any problems getting it to work. And then, lastly, I'll just make sure that all of this reconfiguration has been loaded. So I'll do a sudo systemctl restart httpd and I've just got a typo there. So I'm going to rerun that. So I need to run a sudo systemctl restart httpd and that'll just restart the web server. Okay, so at that point, I'll go back to the console and I'll refresh this main DNS name for the EC2 instance. Now that I've installed WordPress by downloading it and copying into that folder, I can run through the steps for installing WordPress. So I'll select English, United States. I'll name the site, My Cat Pictures. I'll put in a username and a strong password, and then I'll put in my personal email address. Go ahead and click on "Install WordPress." It'll take a couple of seconds, but at this point, WordPress has been installed and it's actually configured the database. It's created database tables and configured the database in the way that it needs. So I can click on, "login" and I can log in using the username and password that I've just defined on. There we go. I'm logged in to a fully functional WordPress site that's configured to use RDS as its database engine. Now, this is everything that I wanted to cover in this lesson. Essentially**, all I've done is configure an RDS instance, installed WordPress on an EC2 instance, and configured the communication between them. But what I wanted you to get out of this lesson is an appreciation for why RDS is really good at what it does. If you've got any kind of relational data for any existing system which uses mySQL Microsoft SQL, Postgres, MariaDB, Oracle, any major relational database product, then when migrating that into AWS, you can consider using RDS.** **RDS is a really exceptional product. It allows you to quickly and easily provision a database as a service and as you'll learn in the following lessons in this part of the course, it offers some really advanced functionality. So it's able to manage backups and restores it can handle resiliency so it can define additional availability zones for these RDS instances to live in and it can also define read replicas which provide international resiliency as well as allowing it to scale from a read perspective. So RDS is a really great product.** So from an exam hints perspective, just make sure that you've read everything on this exam hints and key facts page. Make sure you're comfortable with the different instance sizes. **The fact that RDS is billed based on the instance size based on the provisioned storage, not used storage. The fact that you can have all these different types of instances, the different families, the different types of storage, the different performance levels, the fact that both of these could be changed, the fact that you can use reserved instances, the fact that it can support encryption using KMS.** All of those are key things for the exam, as is the fact that **security is controlled using security groups**. So I've covered everything that you'll need in this lesson from a foundational perspective. Over the upcoming lessons I'll add additional pieces of information on other key facts that you'll need for the exam. But at this point, go ahead mark this lesson as complete and when you're ready, join me in the next.