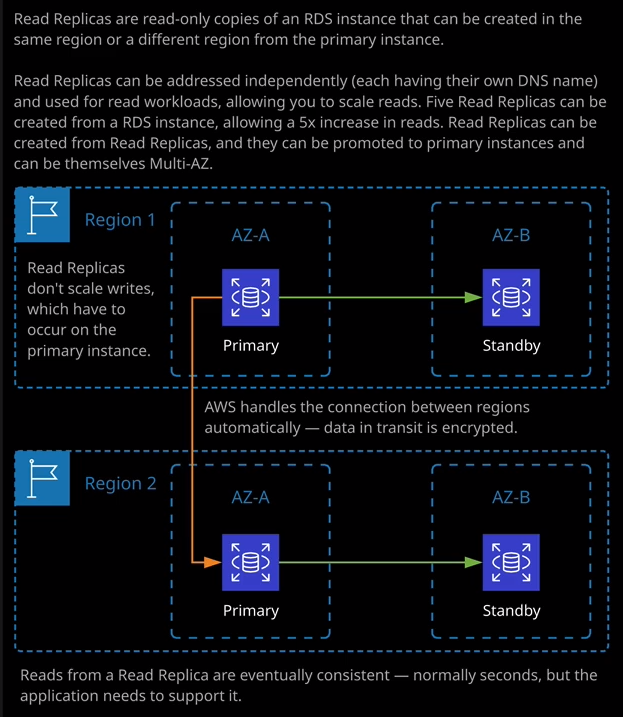
Welcome back and in this lesson I want to follow on from the last lesson where I talked about multi AZ deployments by discussing in this lesson read replicas.

Now **read replicas or RDS read replicas are read only copies of an RDS instance that could be created in the same region or a different region from the primary instance. So read replicas can actually be addressed independently from the primary database instance. So that's how they differ from a multi AZ deployment. You're actually able to address them independently they have their own DNS name.**

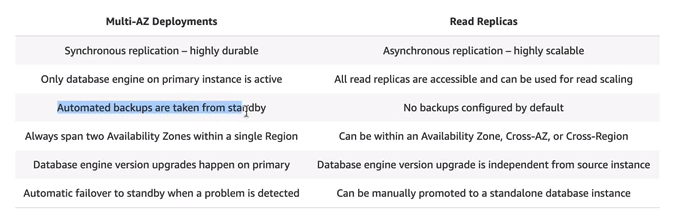


Now to create a read replica, we just select the database instance, go to actions, and then create a read replica and the process is very similar to creating a new database instance. We need to specify the destination region and that destination region can be the same, or it could be a different region and if you select a different region then AWS handles the secure communications between those regions without you needing to perform any networking configurations, that's hugely important either for production usage or for the exam to know that **you don't need to perform any networking configuration to set up a cross region read replica**. Now, in addition to the destination region, you need to specify a destination DB subnet group. Remember from earlier in this topic a subnet group defines which availability zone the database instance, in this case, the read replica is created in. So for this demonstration, I'm going to leave this set to RDS lesson. You can also specify if you've got an availability zone preference or if you want RDS to make that selection for you. So in this case, I'm going to pick U.S. East 1b. A read replica, just like a primary database instance could be publicly accessible or not. For this demonstration, because I'm going to create it in a database private subnet, I'll select no but this could be changed later, just like it can for a main database instance. You can specify whether you want to enable or disable encryption for a read replica. You can also independently specify the instance specifications. So the DB instance class and in this case, I'm going to pick one of the same type, so db.T3.medium. Now a relatively recent addition to this is you can also make read replicas multi AZs so historically a read replica was a single database instance that you could deploy in the same or different regions, but now you can make a read replica itself multi AZ and just like the diagram on screen now, that gives you the ability to replicate the data synchronously between the primary and the standby instance in the primary region. Replicate asynchronously that's an important term asynchronously as opposed to synchronously. So if you see the term asynchronously, then you know that that refers to a read replica, so asynchronously replicate from this primary region to a secondary region or a read replica in that secondary region. I mentioned, that AWS handles the secure communication between regions. If you're using a cross region read replica. If not, then it's in the same VPC and then this asynchronous replication goes to the read replica and if you set it to be a multi AZ deployment, then there's also a standby for that read replica and synchronous replication between the primary read replica and the standby. So you can now use this to create some really resilient architectures using RDS. For this demonstration though I'm not going to pick a multi AZ deployment for the read replica, the read replica source is where it takes its data from. So I'm going to use RDS lesson and I'm going to give it a different instance identifier so rdslesson-rr for read replica. You'll notice that many of these options are similar to creating a full database instance, and that's by design. Essentially, read replica is just another fully featured database, but it's receiving its updates asynchronously from the source from the replica source, which is the database instance you're copying from. So at this point I'm going to go ahead and create the read replica.

Now, an important thing I want you to understand, especially for the exam, **is that in order to create a read replica, the master instance needs to have automatic backups enabled.** This is often one of the key criteria of exam questions which test your knowledge of this. So one of the first steps to creating a read replica is to enable automatic backups on the master instance the source of the replication. Now, I didn't need to do that because I've already enabled it. I was using it in the lesson where I talk to you about automatic backups. So this is yet another reason why you should have automatic backups enabled on any RDS instances. So by enabling automatic backups, you're able to create a read replica. Now, why would you actually create read replicas? That's one of the important things that you need to understand for the exam. Well, there's two main reasons the first is that it is much faster to promote a read replica to be a fully featured database instance than it is to get a snapshot into a database restore. So generally read replicas serve a useful global resilience function. So you can create these read replicas in different regions and then, if you do have a major disaster in your main region, you can promote the read replica and immediately move all of your traffic over to this replica, and that can occur much faster than restoring from snapshot. Now historically, this has been slowed down because you weren't able to use multi AZ read replicas, but that's been resolved now so immediately failing over to a read replica can, if configured correctly, give you access to a fully featured multi AZ installation, and that's really awesome for DR situations. Now, the other reason for creating read replicas is they are independently addressable, but only for reads. So if you've got a database platform that's getting close to capacity, maybe you've got a lot of writes and a lot of reads on this primary instance, and you want to scale out. You can't use multi AZ configurations to allow this because, as I mentioned in previous lessons, you can't address the standby instance. But what you can do is if you need additional writes on an RDS instance, well you can increase the size of the instance. But if you want additional reads, you can scale out. It's horizontal scaling versus vertical scaling, so this is key to understand. If you've got a read heavy workload, let's say that your workload is 80% reads and 20% writes. You can often achieve much better performance by just adding read replicas and then getting your application to address these replicas independently, and that way you can scale reads without having to resort to increasing the size of the primary database instance. Now you can have five read replicas that have a single instance as their source, so this source can have five replicas immediately attached to it but you could also have read replicas of read replicas so you can have a hierarchy of read replicas but be aware that does introduce lag, so that's especially important to understand. So where synchronous replication between the primary and standby is almost instantaneous it is as good as instantaneous. You're not going to lose any data if the primary fails. There can, in theory, **be a small amount of lag between a read replica and the master database instance. So if you are running a read replica, it's asynchronous really important that** you remember that word for the exam. So normally, it's not a huge amount of time but could be seconds. So your application does need to support this eventual consistency model.

So if you're scaling out by adding additional read replicas, your application needs to understand that it could be a number of seconds behind. Now creating read replicas can take some time to do because you're essentially spinning up a brand new instance, so it needs to provision the instance, it needs to establish this asynchronous replication, bring it up to date in terms of the data, and then make it available. But once it is available, it'll have a brand new endpoint address that your application can access directly.

Now, **one limitation of read replicas is there's no method of addressing all of the replicas as one object, so you can't create, say, five read replicas and then have a single DNS name to address all of those read replicas. So your application does need to be aware of your database topology in order to take advantage of these read replicas. So again, another important thing to understand for the exam, read replicas are addressed independently from this main database instance.**



Now I'll make sure this link is in the lesson description but this is an excellent overview of the differences between multi AZ deployments and read replicas. So multi AZ uses synchronous replication. It's highly durable. Read replicas use asynchronous replication, and they're primarily designed for scalability. With multi AZ deployments, the only database engine which is accessible is on the primary instance. With read replicas, all of the replicas can be accessed for read operations. With multi AZ deployments, automated backups are taken from the standby instance so they don't impact performance. With read replicas, you need to explicitly configure backups, multi AZ deployments span two availability zones in a single region. Read replicas, though, could be configured in the same availability zone, cross availability zone, or cross region. So you've got a lot of flexibility about your resilience strategy. With multi AZ deployments, database engine version upgrades happen on the primary. With read replicas, it's independent from the source instance. So you need to be aware of that. You need to make sure that you keep your read replicas up to date. With multi AZ deployments, you've got that automatic fail over to stand by when a problem is detected but read replicas give you that manual control. You can fail over to them when required. Okay, so at this point, after about 20 minutes or so, the read replica has finished creating and I will note that it's got this independent DNS name, so we have to change our application to directly point at this independent DNS address. What we can do, though, is go to actions and promote this read replica to be a fully fledged database instance and when we do that, it stops using this as a replication source, and it becomes a master database in its own right, and it can then be used for read and write operations. So if I go to actions and then promote. I'll be asked whether I want to enable automatic backups. I'll get to select the retention period. I'll get to select whether I want to back window or no preference and if I hit continue and then promote the read replica, it'll take a few minutes to do but this will then be a fully operational database instance in its own right, and that could have been done during a DR event and very rapidly we've got a fully featured database platform just by promoting an existing read replica. So when you're doing this in production, there are some situations where you would create a read replica, even if you don't immediately need one, because you want the ability to rapidly promote it to being a master instance. **Now, do keep in mind that you are charged separately for read replicas. They are their own separate database instance, and so you do incur charges for the read replicas and that depends on the region that you're selecting. The prices are slightly different, but you are charged independently for each one of these read replicas,** and that's everything that I wanted to cover in this lesson. It's been brief, but I do want you to be aware of exactly how read replicas work and their architecture, because it will be important to know for the exam. At this point, we've covered everything that I want to cover about RDS certainly enough for the exam and enough to be getting you started with production real world usage. Now, in the next topic of the course, I'm going to be covering Aurora, which is AWS's in-house developed database engine and it's not just a small evolution over RDS. It's a complete change to the underlying architecture. It has massive performance benefits, massive feature benefits, and I can't wait to tell you all about it. So go ahead, mark this lesson as complete, and when you're ready, you can join me in the next topic, where I'm going to cover Aurora.