Now **DynamoDB has got a number of different ways that it can handle backups and restores.** If I click on the "**Backups" tab, you're able to enable point in time recovery but it's a specific thing that you need to enable. If you enable point in time recovery, you can restore to a point in time up to the last 35 days. That has to be enabled on a per table basis. If you recall the point in time recovery that Aurora was capable of doing or RDS was capable of doing it's the same process, but you have to enable it on a DynamoDB table.**

**In addition to this point in time recovery, you can also create manual explicit backups of a DynamoDB table, so I could do that by creating the backup**. I'll need to select the table name. So it's a per table back up and let's say that I called the back up just back up, keep things simple for this demo and hit "Create" well at that point I've now got a backup of this table in full that I can use to restore. Now, just like with RDS, if I'm restoring this back up—so if I select the backup and then hit, "Restore" I need to restore it to a new table name. But a backup done in this way does store all of the table settings that are listed here, so you won't recognize all of these yet at this point. But it is important to realize that a backup is not just the data, but it's also the configuration, and any indexes that you've got on that table. I haven't talked about indexes yet. That will be coming up later in this topic. But when you need to back up both the data and the configuration, then you can use the built-in backup functionality that comes as part of DynamoDB.

Now historically, encryption was an option that you could enable on a DynamoDB table. But at this point, encryption comes standard. You get the ability to manage encryption and by managing you're able to select between using the service default customer master key or you can use a CMK that you can manage, and if you pick to use KMS, you need to select the CCMK to use. I can't do that because I've got no others defined in this region. But if you do elect to use KMS and your own custom master key then not only can you control the rotation of that key but just like with S3, you separate the role. So it means that just because you're a DynamoDB administrator, you don't necessarily have the permissions to decrypt data in DynamoDB. So that's another really important thing to understand. By selecting KMS encryption, you implement that role separation and you also get a lot of advanced functionality. You can control the rotation of keys and set specific and really granular key permissions. Now I know that we haven't covered KMS yet. We will be doing later in the course, but I did just want to draw your attention to this piece of functionality at this point, it might come up in the exam.

What also might come up in the exam is a feature of **DynamoDB called Global Tables. So by default, any tables that you create inside DynamoDB operate in a specific AWS region.** **You do have the ability though to create a set of multi-master table so you can have tables in different AWS regions that replicate data to all of the other replica tables of that table**. So it's probably easier for me to show you this. If I go to "Create a Table," and I'm just going to create the table. I'm going to call it demo. I'm just going to call the primary key key to keep things simple this is just a demonstration and I'll hit "Create." So this is just a normal table at this point. If I want to enable the global table functionality, then I need to do a number of things first I need to wait for the table to finish provisioning, which it has. **Go to global tables and I need to enable a feature called streams.** I've got a dedicated lesson coming up later in this topic talking about what streams actually are but for the exam, remember **to enable global tables streams need to be enabled on that table**. So that's the first thing, I'll enable streams. Now this starts **off as a single table and to make it a global table I need to add what are known as replica tables** to do that, I'll select "Add Region" and I'll need to select one or more regions that I want to replicate this table to. So for this demonstration, I'm just going to pick one additional region. So I click on "Select Region" and then add Asia Pacific Sydney. It will perform a number of checks and once it's done that I'll hit "Continue." Now this process creates the table and it configures the replication infrastructure between these two tables. Now **it is a multi-region, multi-master replication. So that means that all of the tables are a part of this global table can all be read from and written to. It's not like other replication technology way you've got one master and then lots of secondary tables.** **With DynamoDB, you can read and write to all of the tables and it employs a last writer wins conflict resolution protocol**. **So if you've got three tables and you write to all three of them with the same item then whatever gets written to last wins the conflict and so that data is replicated to all of the other tables.**

Now **you can only enable global tables and tables which are empty**. So that's why I had to create a new table to demonstrate this process. Even though I could go to weather data and go to global tables and enable streams, I couldn't at that point add additional regions because the table needs to be empty when you enable global tables. That's really important to know for the exam I have seen it come up, so it **needs to have an empty table, enable streams, and then add additional regions.**

Now **DynamoDB does come with full integration with CloudWatch, which means you have access to a complete set of metrics about the table and accesses to that table**. Now, a lot of this you won't be familiar with at this point, so you won't know what read capacity or write capacity units are, and you won't know what throttling is. I'll be talking about that in an upcoming lesson, where I'm talking about DynamoDB performance. For now, just being aware that **you do have full integration with CloudWatch** is enough. Now, **in addition to these capacity or performance metrics, we've also got other metrics. So we've got latency, so how quickly you can access data inside DynamoDB and that's for both put and get operations. We've also got metrics on any streams on any TTL items,** which I'll talk about later in this topic on scan and query operations and then any errors. **So you've got access to a full set of monitoring and a full set of metrics that you can use to create alarms if you want to automate any remedial action.**

What I want to finish up with is my thoughts on **when you'd utilize DynamoDB versus other database technologies,** because this is something that comes up really frequently in the exam. **So DynamoDB is a noSQL database. If you see any questions in the exam, which mentioned relational or SQL based data, that's not something that can be easily transferred into DynamoDB. DynamoDB suits unstructured data. So keys and values, keys and then other attributes, or even JSON documents or more complex data types**. In the exam, I'd be specifically looking at questions and eliminating DynamoDB as an answer if it talks about relational data. I'd then be looking at whether the question is asking for a **database product that is delivered as a service.** So as you've seen, there's no mention with DynamoDB about servers or clusters or any of the other terminology that's generally associated with databases. **DynamoDB truly is a database as a service product.** **It's serverless from your perspective, you have no exposure to servers. You can't see servers. You can't control them. With DynamoDB, you're simply making tables and as you'll see in later lessons in this topic, you can either explicitly define performance or configure DynamoDB to manage performance on an on demand basis.** So when you're inside the exam, if the question is looking **for any lightweight database products which don't need to utilize relational data, if it's not a fixed schema, if it's a web scale application, if it mentions ID federation,** which is something we haven't yet covered, they're all keywords that would point me in the direction of **DynamoDB**.