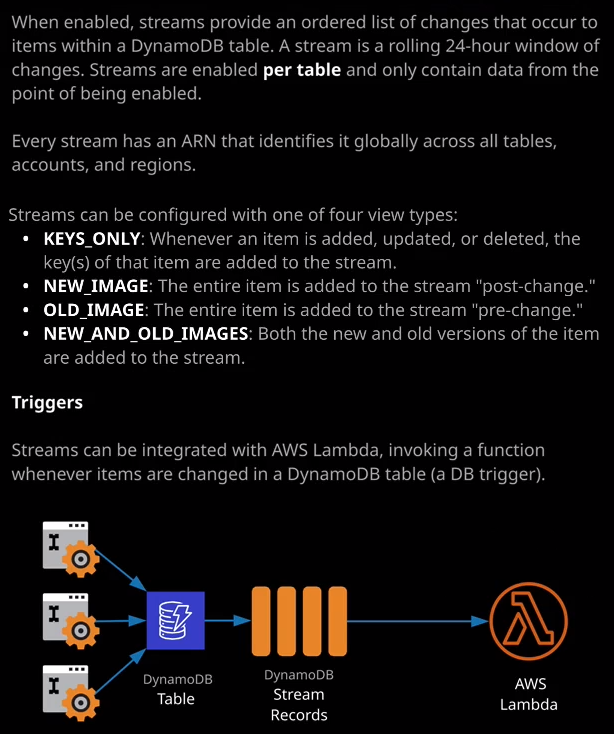
the architecture of DynamoDbBstreams and triggers.



Now **DynamoDB streams are things that are enabled on a per table basis and a stream provides a rolling 24 hour window of any changes to items in that table. So when you create a table and I'll just pick some random text for this new table and go ahead and create it all tables start off with having no stream enabled. So you've always got the option of clicking on the manage stream button and then picking the type of stream that you want to enable on a specific table and all of the different stream types use the same architecture which is this rolling 24 hour window of changes.** It doesn't matter if that table gets one write every six hours or 100 writes every second, the stream will always have enough capacity and enough resilience to store a whole 24 hours worth of changes for every item in that table. So essentially a record gets added to a stream. It stays there for 24 hours and then drops off. Now, every time an item is added, updated, or deleted, to a table which has streams enabled. An entry is added to that stream, which details the insert, update, or delete operation. The information that goes into a stream could be changed by adjusting the view type, and there are four different view types that can be used with DynamoDB streams. The first view type is keys only and with this view, whenever an item is added, removed, or updated primary key for the item. So that's either just the partition key or it's the partition key and the sort key are added to the stream, and that's it. With the new image type, whenever you insert, modify, or delete an item, the new state of the item in its entirety is added to the stream. So by using the new image type, you get an overview of the new state of the item so the post change state of the item. The old image view type is like new image, but it's the inverse. So with the old image view type, if you make a modification to an item, for example then on the stream will be a record of the item before that change, where as new image will be a record of the item after the change. If you pick new and old images then the stream record will contain the item before the change and after the change. Selecting between the view types depends on what you're trying to accomplish. If you pick keys only then it can help you identify when certain items change but you'll need to consult the table to be able to see what's changed but even then, you won't know what the item state was before the change. All you'll see is the current state of the item. The new image and old image view types are great if you want to perform an action based on the item, maybe you want to have it so that when you update an email address, you should send an email confirmation or an email approval so you know how on some websites when you change the email address associated with an account some websites send an email to the new email address, asking to confirm where as some send a request for approval to the old email address. So by picking between new image or old image, you be able to have this change in email address either before the change or after the change. So picking between those two depends on whether you want to act on a change of an item, either before the change or after the change. You don't always have to select, though, because if you pick new and old images, then you get all of the data so you can act on that change without having to consult the table. So that's often much more efficient to store all of that data in the stream because then you could have processes which get the data from the stream, and don't then have to consult the table for additional information. So it's often much more efficient to pick the new and old images view type. Now another really important thing to understand the exam is when you'd use streams; what are streams used for? Well, they're actually used for the replication that's available inside DynamoDB. So when I talked about global tables earlier in this topic of the course, I showed you how you need to enable streams in order for that replication to occur. So streams are used to communicate the changes that are made from one table to the other tables. Streams are also useful to implement trigger functionality so you can define a Lambda function, which can perform certain tasks and then you can have that Lambda function invoked every time new records are stored on the stream. Now, some examples of why you might want to do this is to configure an event driven pipeline to send emails or perform other processes when certain details are updated in a table, maybe you've got a customer order that gets added to a table inside DynamoDB so it gets added as an item and you want to perform notifications based on certain attributes of this order. You might also use triggers to generate summary information. So every time somebody orders or views a particular item of stock. You may want to update a different table in your organization or generate a process, a notification based on that activity. So streams and triggers give you the ability to use event driven processing as part of DynamoDB. So traditional relational databases have had the concept of triggers for a while. So whenever certain values were updated in a database, whenever maybe stock levels go below a certain amount, triggers allow you to kick off a process to replenish those stock levels. Now because DynamoDB is a much simpler databases designed for scalability and simplicity by using streams plus Lambda, you get the ability to implement these triggers in a much more scaleable way because the database itself is not running the compute because it's Lambda and Lambda is highly scalable. You get the ability for DynamoDB to focus on data management and then using streams and Lambda, you can focus on adding event driven processing to any changes of items in DynamoDB. Now again, you don't need to know the intricacies of this for the exam. I want you to be aware of the architecture and just knowing that you can define a Lambda function and have it invoked based on records been added to a stream is enough. So this is the type of architecture that you might see if represented visually. So you got various processes adding items into a DynamoDB table. Every time that happens, you get a stream record that's added to a stream and then a Lambda function is invoked based on these stream records. Now, I mentioned at the start of the lesson you can refer to a stream, and you've always got that rolling 24 hour window of changes. So a Lambda function generally will process records as they're added to a stream but there is nothing to stop applications consulting a stream on going all the way back to the beginning of this 24 hour window or anywhere between that point and the end of the stream. Streams are a rolling window and you've always got access to the records in those streams. Now streams a durable, they're scalable, and they're reliable, so you can treat them as highly available architectures really important for you to understand in the exam. Lambda is a highly scalable solution. Dynamodb is highly resilient, and any of the data records that stored in streams are also highly resilient. So keep that in mind for the exam. Now with that being said, I did say that I'll try to keep this lesson brief. So that is everything that I wanted to cover in the lesson. Now, don't worry. You don't need to be aware of a lot of this stuff from an implementation perspective. So I'm trying to give you just that base level architectural knowledge that you'll need for the exam and enough to get started with, in case you want to do your own background reading about streams and triggers but at this point, that is it for this lesson. So go ahead, mark this lesson as complete, and when you're ready, I'll see you in the next.