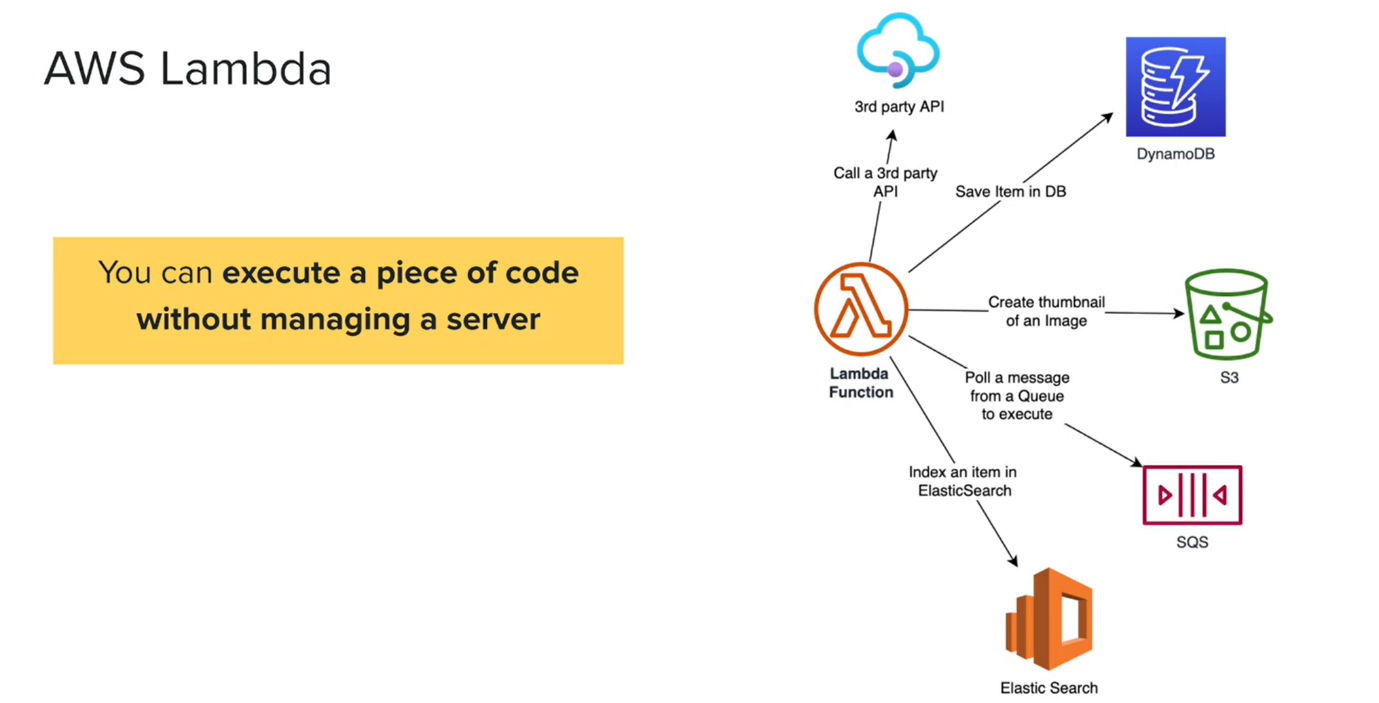
**16. Introduction to AWS Lambda**



* Lambda is the core serverless service in a platform. So, whenever you want to execute a piece of code, you can use the lambda function. Now have a look at this image.

**Fetch information from 3rd party api and store it in dynamodb**

* Let us say that you want to get a weather information from a third-party API. Let us say this is open with the map API and you want to get the weather information in every hour and store it in a database.
* So, what can you do?
* You can create a lambda function and in the lambda function you can have the code to query the third-party API and get the data and save it in Dynamo DB Table.
* if we wanted to invoke this lambda function in every hour, you can write a cron job. So, the cloudwatch cron job will invoke this lambda function in every hour and this lambda function will fetch the data from the third-party API and store it in the Dynamo DB table and let us see another example.

**Create a thumbnail of image**

* Imagine that someone uploaded an image into a s3 bucket. So, you wanted to create a thumbnail of that image. That image may be like in high quality, but you want to create a thumbnail. So, in that case, also you can write a lambda function as soon as an image is uploaded to S3 bucket.
* Invoke this lambda function and create a thumbnail and after that the thumbnail can be stored in a same s3 bucket.

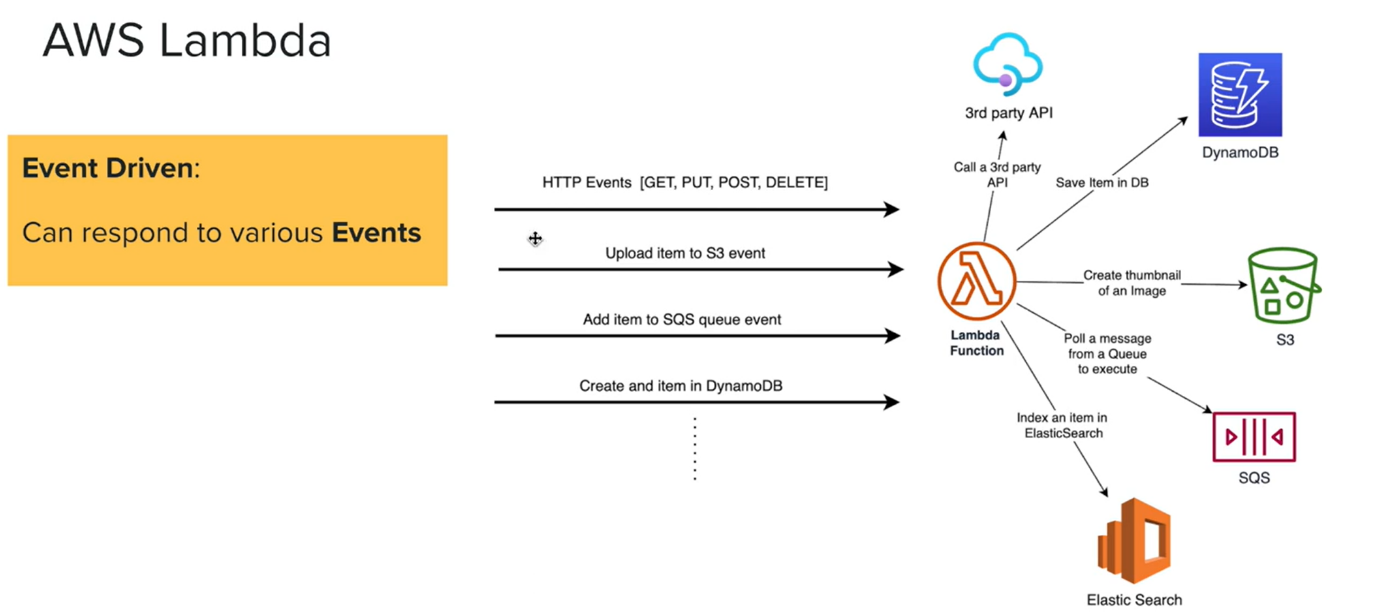
**Poll a message from queue to execute**

* Now here is another example. Let us say that you have a queue. Now, in this case, I am using SQS. So, whenever an item is added to the queue, you can write a lambda function to Poll these messages or these items from the queue and then execute. So, in this case, the lambda work as a worker lambda function.

**Index an item in elastic search**

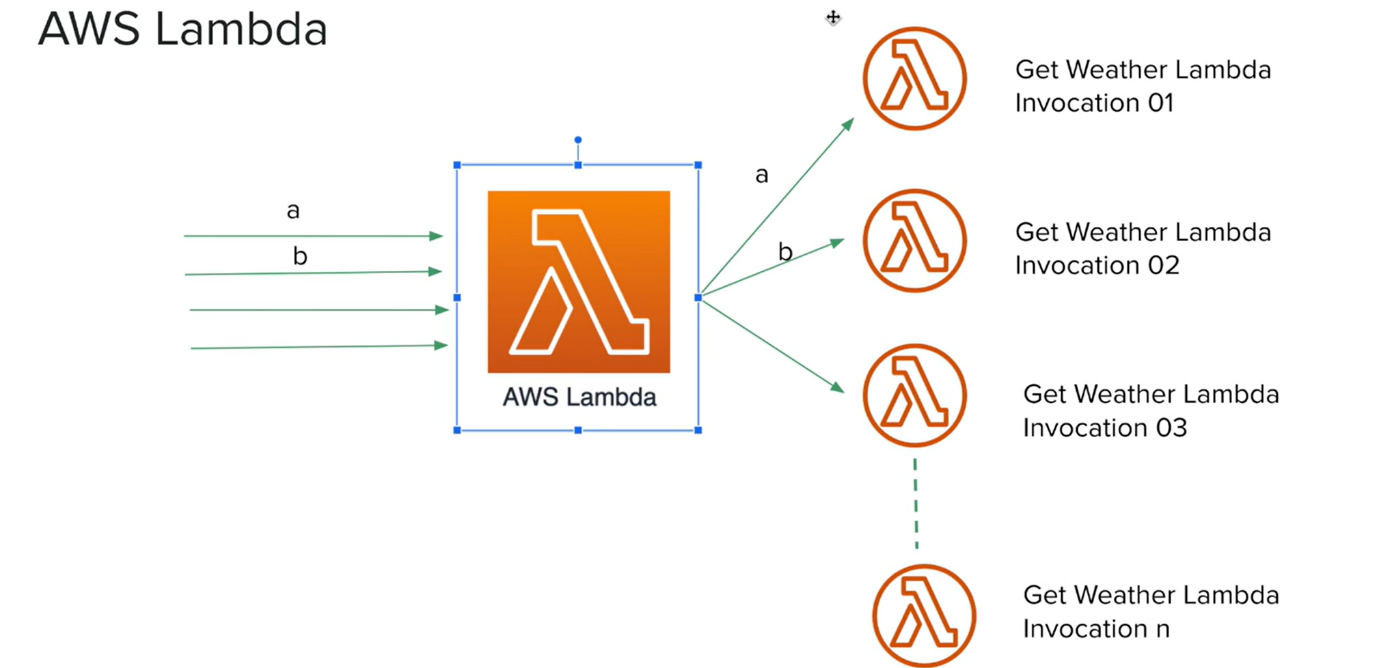
* Now here is another example. Let us imagine someone will add some data into a Dynamo DB table, and the Dynamo DB table will send that data into a Dynamo DB stream. We can write a lambda function to listen to this stream and get this data. Probably we can use it to index it in a Elasticsearch.
* Some of you might not be familiar about Dynamo DB streams or Elasticsearch, but the idea I want to convey to you is that a lambda function can be any piece of code that you want to run without managing a server and this piece of code can be invoked by any event.
* So, the next important point that you should remember about Lambda function is that they are event driven and we already discussed so many different events and, in these slides, I'm showing you some of these events.

**AWS lambda events**



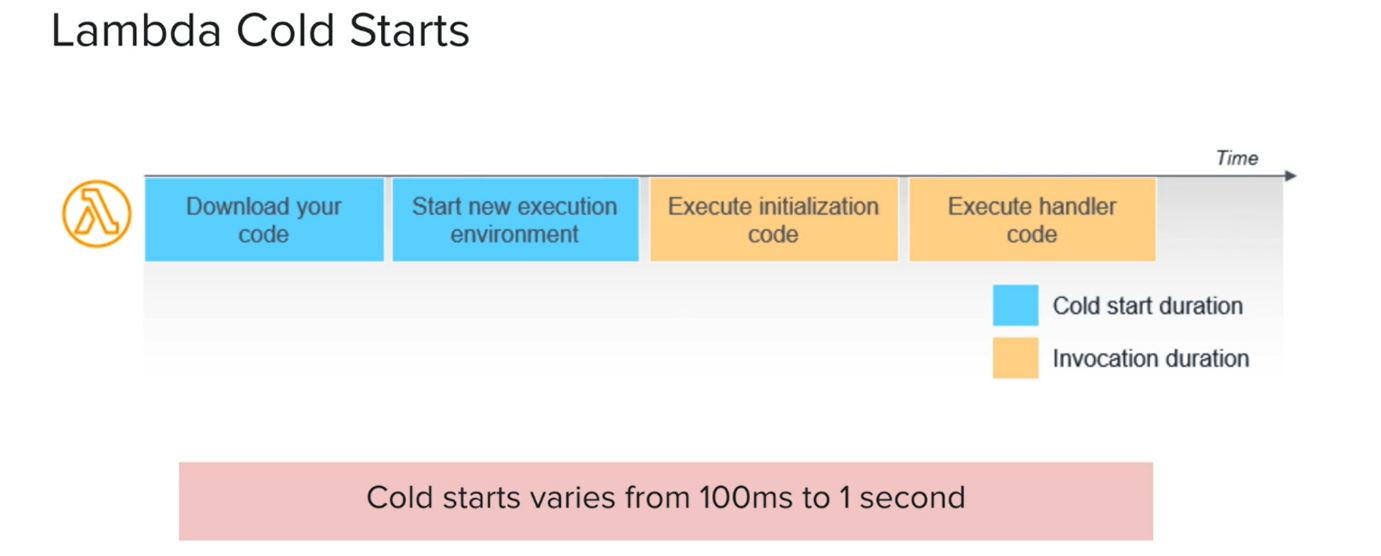
* One common event is this HTTP event. Now, these are standard HTTP events like get, put, post delete and this standard HTTP event, we can invoke lambda functions through these events and this is a popular use case and there are other events such as an item is getting uploaded to an S3 bucket.
* We already discuss it whenever something is added to a s3 bucket. We can use that event to trigger a lambda function in this case to create a thumbnail of that image.
* it could be like adding an item to an SQS. So, in this case, the Lambda will poll that message from the SQS queue and execute it.
* then we already discussed the Dynamo DB streams. So, whenever an item is added to the Dynamo DB, through that streams we can call upon a lambda function and that lambda function can be used to maybe index that item into Elasticsearch to optimize searching in your applications
* lambdas are event driven. We will not have lambda functions running long time. It will only respond to an event and then it will go away and the next point is that the lambdas are serverless. What does serverless mean. Essentially two things. You only pay for the usage. In this case, you pay only for the execution time of your Lambda Code and then you do not have to manage any servers. AWB will manage those servers for you.

**Important things we should remember about aws lambda**

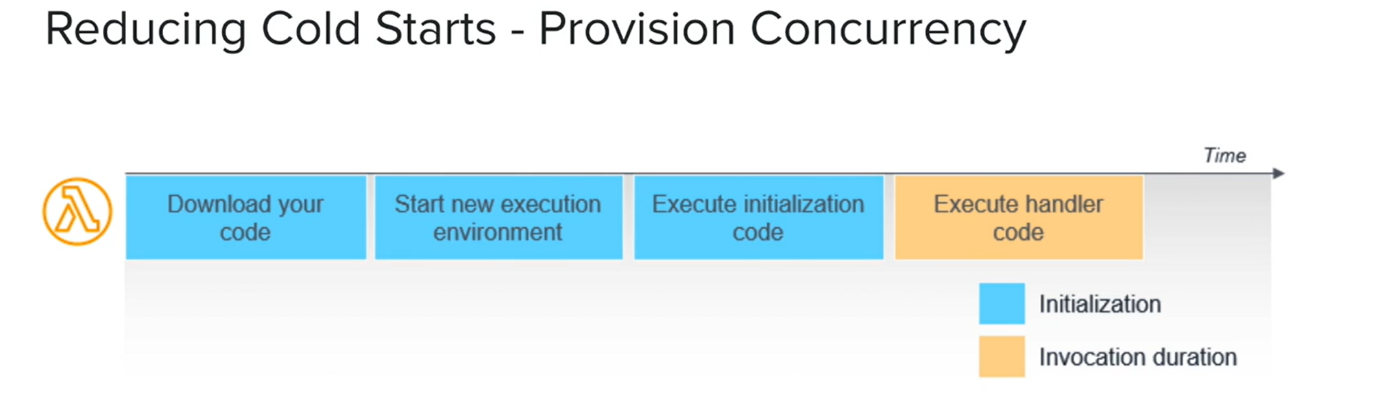


* here is another important thing that you should remember about Ada Lambda. aws Lambda is the managed service. Aws Lambda and lambda functions are two different things.
* When we get a lot of requests to invoke a particular lambda function, aws lambda is the managed service that will handle those requests. Now have a look at this example. Let us say we get so many requests to invoke this get weather lambda function.
* So, if these requests are coming in parallel, AWS Lambda will invoke lambda functions in parallel. So, it is the same code, but it will create parallel lambda functions and there is also this concurrency limit per region. You can have up to 1000 concurrent lambda functions running.
* Now there is another important thing that you should remember. Let us imagine that this **a** request will invoke this lambda function. So, this is the invocation **a** and the same time, another **b** request wants to invoke the same lambda. So, in this case, **a** lambda will never send the same request to the **A** invocation. It will always create another lambda function. Now this is to prevent a particular request context data leaking to another invocation.
* however, these lambdas are running in execution environment. Now you can consider these lambda functions as containers. Now these containers can be shared among these lambda functions.in order for B lambda invocation to use the execution environment of **a** lambda function, this **a** lambda function must complete and after that is completed, if there is a B request, then a lambda will invoke the second lambda function in the previous execution environment and we call this a hot start.

**Cold and hot start**



* So, let us talk about cold start and Hot Start a little bit in detail.
* Now when you use lambda functions, there is something that you should be concerned about and that is lambda cold starts.
* So, what is a code start?
* Have a look at this diagram for AWS blog. Lambdas are invoked by an event. So, when there is an event, then lambda will invoke a lambda function. Now before the lambda functions getting invoked, aws lambda must download the code for that lambda functions, and after that it must create an execution environment with the runtime. It could be Node.js, C-sharp, Python, whatever. The runtime that we associated with the lambda function needs to be started in that execution environment so, you can consider this as a container and after that it will execute the initialization code and finally it will execute the handler code, which is the code of the lambda function that we want to execute. Now the time that it takes for the first two steps is called cold start duration. Now, this could vary from 100 milliseconds to one second and although you are not being billed for this duration, it will add down to your latency of the lambda functions.
* However, the subsequent invocations of the lambda function and if they happen quite frequently, will try to reuse the same execution environment. So, in that case we call them hot starts because the code is already downloaded and the execution environment is already available.
* However, remember that AWS Lambda will take out this execution environment time to time. So, there is a possibility that 1% of your Lambda invocation could be landed on cold starts.
* So how do we prevent this?



* Now, there are multiple ways that we can reduce the cold starts. So, one of the recommended ways is to use provision concurrency. Now, this is essentially we will keep some execution environment already created for our lambda functions. So, the code is already downloaded and the execution environment is already available and the initialization code is already run.
* So whenever there is a request, it only must execute the handler function. So, the cold start will be minimized. However, when you add a provision concurrency, you must pay extra for that.
* So that is about AWS lambda and we are going to use a lambda quite a bit in this course.