

Welcome back and in this lesson I want to talk about a specific feature of CloudWatch, known as CloudWatch logs. So in the last lesson I talked about how CloudWatch was responsible for gathering data points and metrics from both AWS services as well as custom sources. Now, CloudWatch logs is a specific subset of CloudWatch that's designed to collect and store log data. So rather than storing data points and metrics, CloudWatch logs is designed to store log events. So if you look at the diagrams on the right of my screen moment. Essentially, the base entity of CloudWatch logs is what's known as a log event and a log event at its foundation is essentially a timestamp. So year, month, day, hour, minute, second, and that, together with the wrong message, makes up a log event and log events are grouped into a log stream. So a log stream is a sequence of log events with the same source. A log group is a container for multiple log streams, and it's on the log group that you can configure retention, monitoring, access control, as well as metric filters that I'll talk about later in the lesson. Now you can already see that I've got lots of log groups that are already present in the account that I'm using to create this lesson. So, essentially, whenever you've got any product or service that's designed to put its log data into CloudWatch logs, you'll tend to see that it will go ahead and create its own log group. Now any AWS services generally prefix this with /aws and then the product name and then slash and the particular entity inside that product. So you'll see that with Lambda, it's AWS Lambda and then the function name. So we've got a couple of functions that I've used while making this course. So the Lambda human approval I used during the hands on lab, where I demonstrated cat slave 9000. Now that we've got another function that was used in that same demonstration, a Lambda function that created some image thumbnails, and then another one that I was using for testings or all of the Lambda related functions. They're all grouped up into a log names that are prefixed the same, but essentially, whenever you have a service or a product or a particular application, sending data into CloudWatch logs the log group is a thing that it will create. Now later in this topic, you going to get some experience inside a hands on lab of using the CloudWatch agent and as part of that, you'll be grabbing some application logs off an EC2 instance. Now if you're using the CloudWatch agent to grab particular application or operating system logs, what it will tend to do is create a log group for that particular log, maybe var lock secure or var log messages. Inside that log group will be individual log streams, which represent each of the EC2 instances that are capturing that log file. So you might have three instances that each are capturing var log messages and so the log group will be called var log messages. The log stream will be named after the EC2 instance, and then inside that log stream, you can have individual log events that are relevant to that particular instance for that particular log file, so you'll see that demonstrated in the hands on lab but CloudWatch logs essentially is used for lots of different functionality across the AWS products and services so things like VPC Flow Logs that you'll see later in this topic, CloudTrail that you'll see later in this topic, they can all put log data into CloudWatch logs. Something else that could do that if I move across to the all services part the console and go to Lambda. I'm going to demonstrate this by creating a really quick Lambda function. So I'll create function. I'm going to author from scratch, going to change the running time down to Python 3.6. I'm going to call the function logging test just to make it easy to identify. Going to expand this box and I'll need to create a new execution role for the Lambda function because it's the execution role that will give the Lambda function the permissions to write that data into CloudWatch logs. So that should be a standard set of permissions that anything has if it uses roles for its permissions inside AWS, you should always give it permissions to write to CloudWatch logs. So now that I've got that set up I'm just going to create the function. I'm going to leave everything as default because I actually don't care what this function does. What I want to do is just test the function, so I'm going to hit test, and I'm going to use the event template of hello world. I'm going to call the event name hello and hit create hit test again. That's just going to execute that Lambda function. I'll just expand details. We can see that that it did execute. Everything looks good. Now that it's executed, I could move back to CloudWatch and select CloudWatch logs and will note that we've now got this new log group the logging test Lambda function. If I go into that log group because it's Lambda function, you'll see a log streams for various different groupings of time. So where EC2 uses the log streams for individual EC2 instances, Lambda will group it based on execution times. So I'm going to go ahead and click on this log stream and what we'll see is just a piece of sample output for this one execution. So it's a simple as that as long as the AWS service is running with an execution role or another permissions role that gives it the ability to write into CloudWatch logs, then it can go ahead and create a group so a CloudWatch log group in there, it can create a log stream, and then it can enter its logging events into that log stream. So there isn't really much more to say about the mechanics of CloudWatch logs, certainly not for the associate level of the associate essay exam. Essentially, you just need to know the architecture that there are log groups, inside log groups are log streams, inside log streams are log events, and then a AWS services can add stuff into CloudWatch logs but equally you can use the API. So if you are a developer and you writing an application, you can have it directly write out its logs to CloudWatch logs. You can also use the CloudWatch agent to pick up operating system or the application logs, and you'll get some practical experience of that in the hands on lab. Now it's the log group where most of the interesting configuration happens. So if I go to actions, I'm able to take a log group and export all that data to S3 as well as streaming any new data to either AWS Lambda, or Elasticsearch. So they're two very important things to understand for the exam. It is on a log group that you configure export settings as well a streaming settings into other AWS services. You also configure expiration settings on a log group basis, so I can select that and change the retention to a value that is appropriate for every log group is. By default, it's never expire, so the data remains there indefinitely, but you can set it to expire after a configurable amount of time. What you're also able to do is to click on metric filters and add a metric filter. Now what this does is it does tech searches or pattern searches over all of the log data that's in a log group and you can define certain patterns. So, for example, if I wanted to look for the occurrence of the word duration than I could copy duration and paste into this box and then test the pattern, what that would do is it would show us any items in that log where the name duration was mentioned. Now, what I could do with this metric filter is use that to create a metric. So I could call this metric duration, and it would show every time the word duration was mentioned in this log group. Now a more realistic use case for this is you might want to monitor a log file such as var log messages or var log secure or the Windows equivalent, and look for any failed connection attempts, you could do a search in that log group for the text that you see if there's a failed connection attempt. You might be looking for a certain piece of text that your application generates if it experiences a failure and what you can do it, you can use that to generate a metric inside CloudWatch. So if we wanted to use this as an example, so the filter is duration and we wanted to create a metric name. Let's, for argument's sake, just create one called Lambda duration test. So the name doesn't make any sense. That metric doesn't make any sense but we are just using this to demonstrate the functionality. So I'm going to go ahead and create this filter, I have created the filter, and that means that we've now got a brand new metric called Lambda duration test, and we can go ahead and create an alarm on this metric as with any normal metric. So if we wanted to create a metric filter looking for failed SSH or RDP connection attempts. We would get data inside that metric every time that occurred and we could create an alarm that went into an alarm state any time it saw that and based on that alarm, we could take one or more actions. So this is a really powerful feature of CloudWatch logs. You'll definitely use it if you use AWS in production but you might also face exam questions which ask you how you create a metric filter. For the exam, remember, it's created on a log group, and it pattern matches text, and it creates a metric. From that metric, you can create an alarm. Now why this is actually really useful is because CloudWatch logs supports integration with lots of other services so EC2, on-premises servers, Lambda, CloudTrail, Route 53, VPC Flow Logs, custom applications that you might design either by having it directly talk to the APIs by using CloudWatch Agent to look at the log files. There are a lot of possibilities why you might want to ingest log data into CloudWatch logs and if you do ingest log data, you'll likely want to create metric filters and alarms based on those metrics but with that being said, that is everything I wanted to cover around the theory of CloudWatch logs. So go ahead and mark this lesson as complete and when you're ready, join me in the next.