Okay so before we finish up this lesson, I did just want to give you a really brief demonstration about how SQS works. So I'm going to move across to the SQS part of the console, and I'm going to create an SQS queue. So we're going to get started now I'm in the U.S. East North Virginia. I'll call the queue SQS lesson, and I'll make it a standard queue, and then I'll just click on the button to quickly create the queue because I don't need to worry about any of the advanced configuration. So SQS queues just like SNS topics are able to have resource policies so you can define policies on the queue which controls who can add messages to that queue. You got a full set of monitoring as part of a queue. So CloudWatch is able to react to messages that are on the queue and because these metrics are available, you can configure an auto scaling group to react to the number of messages on the queue so this is how you can implement this scaleable backend architecture. Now as well as EC2 instances being able to react to messages on the queue, you're also able to have Lambda functions that are invoked based on messages arriving on the queue. So if you've got any exam questions or real world production usage, **where you need to react to messages getting added to a queue and the processing time for those messages might be less than the maximum runtime of Lambda then Lambda is always going to scale much better and much quicker than new auto scaling groups**. You should think about Lambda if you do have any exam questions that talk about rapid and agile scalability.

Now, the way that you referenced SQS queues from the command line is using the URL. We do not use the ARN so I'm going to copy that URL into my clipboard and then move across to my terminal windows. I've got two turn ALS set up side by side. I want you to imagine that the green terminal on the left inside is the source, So this is the one that we're going to use to add messages to the queue and then the blue terminal on the right hand side is the destination. So this is the side that we're going to use to poll the queue. Now I want to get information about a queue I can use AWS SQS get-queue-attributes that I'll need to specify double hyphen queue-url only to put in that unique URL for the SQS queue and then double hyphen attribute hyphen names and then space all and if we're on that command, I will receive a full configuration overview of the SQS queue. So some important values on the queue. This is the approximate number of messages, and this will give you an overview of approximately how many messages are currently on the queue and this is a metric that you'll tend to use for auto scaling groups. So this is the one that you can use to scale based on how many messages are in the queue. We got the approximate number of messages not visible, so this will give you an overview of the number of messages that are potentially currently being processed. Remember when I said you can poll a queue. The message is hidden for that visibility time out period. Well this metric will show any messages that are currently hidden. **Now the visibility timeout period by default is 30 seconds, but you can set it anywhere from zero seconds to 12 hours.** There's also the retention period, which is important. **So every queue has a message retention period and this is a period that a message can stay on that queue before it's discarded. Now the default is four days and this could be set anywhere from 60 seconds to 14 days.** Okay, and to send a message to the queue, I'll use AWS SQS send-message. I'll need to specify the queue URL so --queue-url and then the unique URL for this queue and then space, double hyphen message, hyphen body and then the payload of the message. So I'll just say hello world, and that will send that message to this SQS queue. Now remember, there are two ways that can perform a poll on a queue a short poll or a long poll. A short poll doesn't wait for messages. It checks the contents of a queue and it retrieves any messages on the queue either one message or up to 10 messages but it doesn't wait around for messages to be added to that queue. The way that we perform a short poll is with this command. So aws sqs receive-message and then the queue URL and we can see that we've got our hello world message that's returned by doing the short poll. Now, if I do another short poll immediately, what we'll find now is we don't get any messages returned from the queue because that message is hidden for the visibility time out of 30 seconds. I can keep running this poll command on because it's a short poll it won't be waiting around. It'll keep returning. zero results and then, once that visibility timeout was expired I'll again, get this message returned. So this is the architecture of a short poll. It doesn't wait around. You keep running it and it's simply returns zero results if the queue is empty but I want to stress this architecture of how a message that's received is not deleted from the queue straightaway. Essentially, it's hidden for this visibility timer, so I'll just clear the screen to make it easier to see. I want to compare and contrast that short poll operation with this command. Now this is a long poll what this command does is perform the same basic operations as a short poll but this time it's going to wait 10 seconds, and it's going to allow us to retrieve a maximum number of messages of 10. So essentially, this will wait around for a number of seconds. If there's no messages on the queue it will wait for any messages to be added to the queue and then, once the time period expires, it will return up to 10 messages. So if I press enter, we've only got one message on the queue and so it's going to return as that message but if we run it again, it is going to sit and wait for further messages to be added to the queue. So a long poll will always return any messages on the queue at that point, or it will wait around for this wait time in seconds. Now if go back to this source window and I send another message to the queue this long poll will immediately return. So it waits for any messages getting out into the queue and then it returns those messages immediately. Now, a long poll, as I mentioned earlier in this lesson, is much more efficient than a short poll. A short poll remember, will return even an empty queue. So you'll have to keep running it over and over again to wait for any messages. A long poll though, will wait until it receives at least one message. So you're not going to get this constant API calls returning zero messages. Okay, so before I finish up this lesson I want to demonstrate the process of permanently removing a message from a queue. The first step is all need to actually poll the queue and I'm going to use the short poll operation. So this is a short poll command and watch when I receive the message, it will contain a field known as the receipt handle and the receipt handle is a unique ID for this message, and I can use the receipt handle to permanently delete the message. I'm going to go ahead and do that. So there we've got the message, and it's got the receipt handle in. Next I'm going to run this command but I'm going to copy the receipt handle from this poll. I'm going to paste it in and press enter, and that's going to permanently delete this message from the queue. So this was a short poll operation. It received the message from the queue. The message would have been hidden for the visibility timeout. The receipt handle was provided as part of that message and I use that receipt handle to delete the message permanently from the queue. So now if I just clear my screen and tab up to the short poll operation. Now we get another message because remember, I sent two messages to this queue. So I'll need to follow the same process to delete the second message. I'm going to get that command again. Copy the receipt handle again, paste it in and delete that second message. Clear the screen again. I'll run another short poll and this time we get no results returned because we've got no messages on that queue anymore. The two messages that we sent to the queue and the terminal on the left of the screen we deleted both of those permanently. So I'm stressing this because it's important for the exam, you will probably find that you will get a question which revolves around how messages are handled on an SQS queue. I want you to be aware that when you send a message to the queue and then you poll that message, it only gets hidden. It doesn't get permanently deleted. Once the visibility timer expires the message reappears on the queue and you can again do a poll and receive that same message. **To permanently delete it you need to use that receipt handle and delete the message permanently.** That has been a fairly command line heavy and complicated demonstration, but I do think it will help you in the exam to understand exactly how this queue architecture works. Strictly speaking, though, for the Solutions Architect Associate exam, the main points you need to understand is exactly how a queue works. You add messages to it. You could retrieve messages from it using short polls or long polls that will come up in the exam. Please remember the difference between a long poll and a short poll. **A short poll returns immediately whether there are any messages or not. A long poll if there are no messages it will wait until a message is added to the queue and then return that.** Remember ques can support **resource policies so you can use permissions policies on the queue itself to control access. It includes a full set of metrics that are available as part of CloudWatch, and those could be used to scale on Auto Scaling group based on the number of messages on the queue. It supports encryption and as well as using auto scaling groups and EC2 instances you can also use Lambda functions which get invoked whenever messages get out into the queue.**