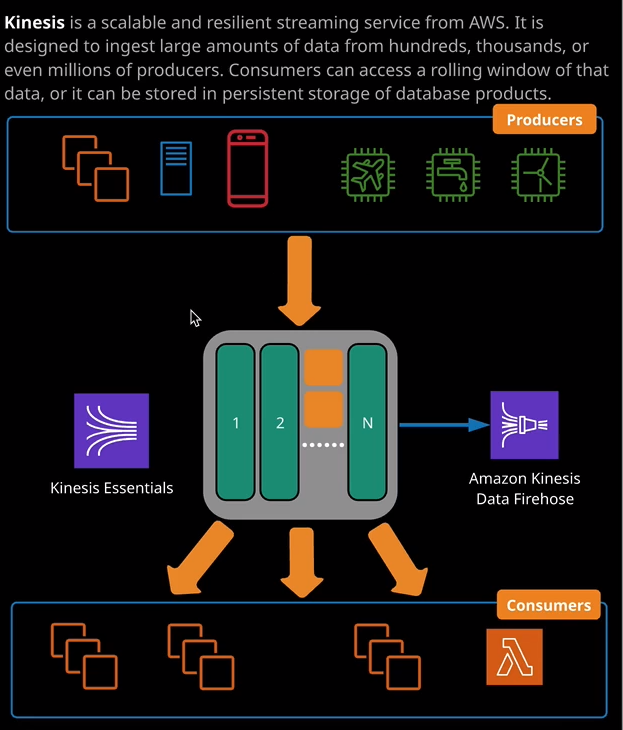
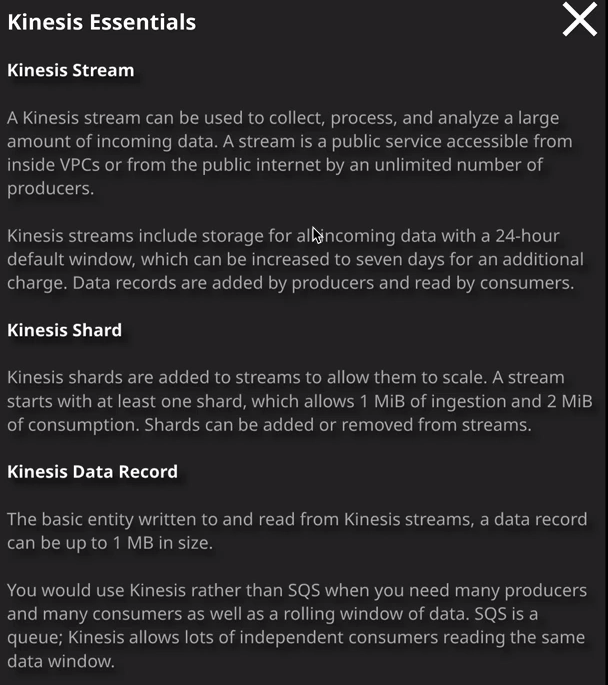
Welcome back and in this lesson I'm going to cover another product from a theory perspective, and that's Kinesis. Kinesis is another tool which for the Solutions Architect Associate exam, you need to know it exists. You should be comfortable with the architecture, what it does, when to use it, and how it's different from SQS. Almost all students in your position start off being relatively confused on the differences between SQS and Kinesis and if I have one priority in this lesson, it's that my students leave the lesson being perfectly clear on how Kinesis is nothing like SQS they're both completely different products which are used in different architectural situations. Now apologies in advance again to keep this efficient Kinesis is another theory-only lesson. For the exam at an associate level, you don't need to know anything about the products implementation details, so we're going to keep it to the theory and architecture.

So **Kinesis is a product which is designed to allow you to process realtime data. Imagine a situation where you have millions of IOT devices, all sending telemetry to your on premises IT location. The amount of bandwidth that you'd need, the amount of compute to process that incoming data, API endpoints to manage that data and storage to keep that data stored persistently when it arrives are generally barriers to entry for using large scale streaming platforms.** Imagine that you've recently developed a mobile application and the application has gone super viral. The application tracks whatever a user taps, which areas of the application are most popular and all of that information is streamed. When you had 100 users, you could handle it but what about 100 million? **Kinesis is designed to ingest huge quantities of data in realtime. It's a fully managed service, and it's one of the most scaleable products in AWS. It doesn't really have a maximum performance.**



So this, at a high level, is how Kinesis works. This is its architecture. At the top, we've got producers things that put data into Kinesis. This could be IOT devices, EC2 instances, on premise servers, mobile devices, these could really be anything that's capable of using the Kinesis APIs.



At the bottom, we've got the consumers and consumers within Kinesis are any entities which can consume the data that's been added to Kinesis and more on this in a second. In the middle, we've got the Kinesis stream, which is the basic entity of Kinesis. Now a stream is what you put data into. It's the service provided by Kinesis which includes capacity to ingest data from producers and allow consumption by consumers. A stream contained storage for a rolling 24 hour window of whatever data volume producers inject into the product. Now this could be 24 hours for 100 producers, or it could be 24 hours for a 1,000,000 producers. Whatever the data volume, you've got enough storage for 24 hours worth of that data. Now, don't confuse this rolling window provided by a stream with the queue. The SQS has a queue has messages which added to that queue and then they get processed and it's part of that processing step there permanently removed from Mac, you know, when that occurs, there gone a queue is not persistent storage. A stream provides a rolling 24 hour window. Nothing is removed and nothing is altered until the end of this rolling window. When a data record reaches the end of this window, it drops off, but until then it persists within this window, and this 24 hour window could be extended to seven days for an additional cost. Now, if you remember from the DynamoDB streams lesson, you might think that a lot of this sounds familiar, and that's because it uses the same underlying technology. A stream can scale almost infinitely, and it does this with what's known as a Kinesis shard. A shard is what provides the capacity for a stream. Each shard gives one mb of ingestion capacity and 2 mb of consumption capacity per second. So if you need your stream to be able to cope with additional load then you simply and more shards. By default, all producers share the ingestion capacity and all consumers share the consumption. So if you need more data rate, or you need to be able to cope with more consumers and producers then add more shards and the data that producers add and consumers read. It's called a Kinesis data record, and they could be up to one megabyte in size.

So let's just quickly move across to the Kinesis console and I'll just show you exactly how this looks from a console perspective. Kinesis is actually supports a number of different subproducts. So we've got Kinesis data streams and Kinesis video streams but for the Solutions Architect Associate exam I am simply going to be focusing on data streams. So I'm going to click, get started. I'm going to create a data stream. I'm going to call the stream test just to keep things simple. So this is the architecture and visual form. We've got producers on the left writing data into a stream, so data records into a stream and then consumers on the right reading that data and in the middle, inside the Kinesis stream, we've got these shards, and it's the shards that influence the capacity. So if I select one shard then I'm able to write one mb into the stream per second, I'm able to read 2 mb from the stream per second and both of those shared across producers for writes and consumers for reads and there's also a maximum of 1,000 records per second if I increase that to two shards then that doubles. So there's a linear relationship between the number of shards and the performance so you could increase the number of shards to whatever level you require for the load would in terms of producers or consumers and, of course, those shards can be updated after the stream is created, so you've got the ability to scale in or out as required. So I'm just going to go ahead and create the stream. That will be the limit of what I'm demonstrating though, because to use a Kinesis stream is well beyond what you'll need for the exam but essentially, a Kinesis stream is given a stream ARN. It takes a few moments to configure. Once it's configured, it's accessible. It includes all the storage required to store any of the incoming data for that 24 hour period and, of course, as I said, that can be updated anywhere from 24 hours to 168 hours. So you've got some flexibility there. It supports server side encryption and integrates tightly with CloudWatch. You've got access to all of that monitoring and metrics. Now this 24 hour rolling window is the big difference between Kinesis and SQS, the rolling 24 hour window, the ability to access all of the data that's produced and added to the stream is what separates it from SQS. SQS is conceptually designed to have entities adding messages at one side of the queue and one entity reading a message at the other side, processing that message, and then removing it from the queue. It doesn't support one message being processed by many entities. Remember, in the application integration topic of the course, I talked about the architecture of SNS and SQS and I demonstrated this architecture. You could have one SNS topic, and to that topic will be subscribed multiple queues and this would allow you to add one message to a topic, have that message cloned to multiple queues. So now each of these queues would have that same one message independently in each of them, and that would essentially give the ability to process to or more versions of that message and the scenario I talked about was an example of where you'd upload one raw video file to a video processing website and have multiple different sizes and bit rates generated from it. Now to be honest, it's kind of a hack. It's kind of a hack that allows one message to be cloned and processed multiple times.

The **thing with Kinesis is that it's not a queuing system. You don't use this to decouple applications. You don't use it for inter process messaging. What Kinesis is for is to allow the ingestion of large scale, real time data. It takes that data in, its stores it in this rolling window, whether that's 24 hours or anywhere upto 168 hours but then at the consumer side you can have these consumers all reviewing and processing each day to record independently**. So let's say that it's time series data that's produced from IOT sensors. While you could have some consumers that reviewing every single day to record, you might have some consumers that are only concerned about values on the hour. Maybe they're only wanting temperatures on the hour every hour. You might have some consumers that only want to check every fourth hour or every six or every 12. **With Kinesis, you have the ability to have consumers, which are consuming the data in different ways, and crucially, they're not touching the data that's on the stream. They're not changing it. They're not modifying it.** S

o that's the architectural difference between queues and streams. Queues are used for messaging. Streams are used for large scale, real time streaming. Now what makes Kinesis really cool is a side product called the **Kinesis Data Firehose and strictly speaking, it's a separate product, the Kinesis Data Firehose can accept records from a data source so it could be directly connected to buy producers, it can take that data and in its simplest form, it can persistently store that data on to S3. So the Firehose offers you the ability to persistently store this large scale streaming data but as well as being able to store it into S3 it can also be stored into things like Red Shift, ElasticSearch clusters, or it could even be stored in third party products such a Splunk but it can also perform data transformations on this live streaming data so you can write sequel like queries actually have some effect real time streaming data that's accepted by the Firehose before it's stored in an S3 bucket. Now, what makes Firehose really awesome is the fact that it can actually tightly integrate with a normal Kinesis data stream. So rather than having to have these producers directly talking to a Firehose a Firehose can attach to a Kinesis stream and essentially take all of the data out of that stream as it's added, ingest it into the Firehose, and then persistently store it in S3 and this is a really effective way that you can cheaply store all of your streaming data records onto a persistent data store without needing any consistent hardware.** Remember all of these are delivered as a service, and you're only being billed for what you consume. So it's a really powerful set of features, so Kinesis and Kinesis Data Firehose. So just in summary for the exam, don't confuse Kinesis with SQS. **SQS is used to decouple things for inter process into service or inter server messaging, and to allow for decoupled auto scaling architectures. Kinesis is used for massive scale data ingestion, consumption, and using Firehose to store that data persistently**. Now, **Kinesis is a service you're billed per shard hour, and there's also a charge for every 1,000,000 payload puts, and a payload put is 25 kilobytes of data,**