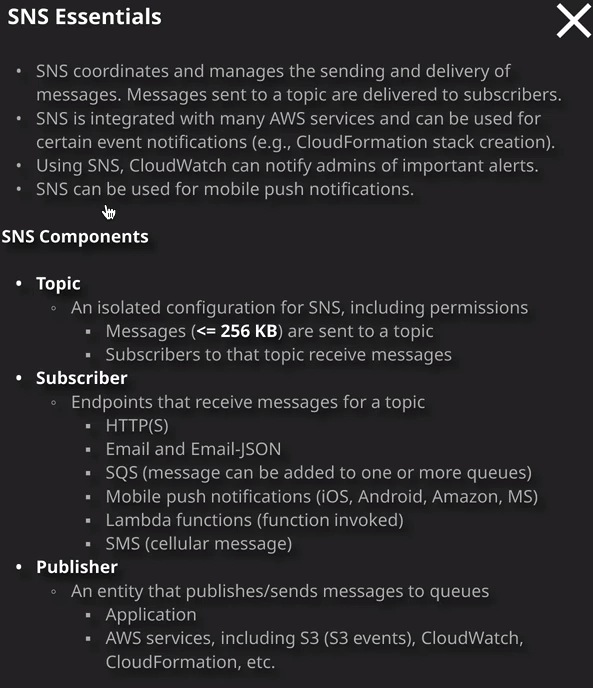
application integration services.   
I'll be talking first about SNS or simple notification service then SQS the simple queue service, and lastly, Elastic Transcoder.

So SNS or the simple notification service is, in my opinion, one of the best and most underrated services available within AWS. **SNS is a publisher subscriber based service**. So the base entity within SNS is a topic. Various different entities can send messages to this **topic**, so it might be AWS services such as CloudFormation, CloudWatch, EC2, auto scaling groups. It could be applications that you've designed and use the APIs to send messages to the topic. You could be using the CLI tools. You could be using the console. You could be using a third party monitoring solution. There's lots of scenarios where you're going to have a need to send messages to an SNS topic.   
Now at the other end of a topic are **subscribers** and when a message reaches a topic, that message is sent to all of the subscribers and that's basically the architecture and I'll be talking through what types of wider architecture SNS supports as we go through this lesson but first, let's just cover the basics.   
Now publishers to SNS topics could be almost anything. I have mentioned a few of them a second ago. So CloudWatch can publish alarm notifications to SNS. CloudFormation can send event updates when stacks are created, updated, or deleted. EC2 can send notifications to SNS, custom applications could do it, auto scaling groups could do it, third party monitoring solutions could do it, almost any service within AWS that needs to send any notifications or do any user notification will be using SNS.



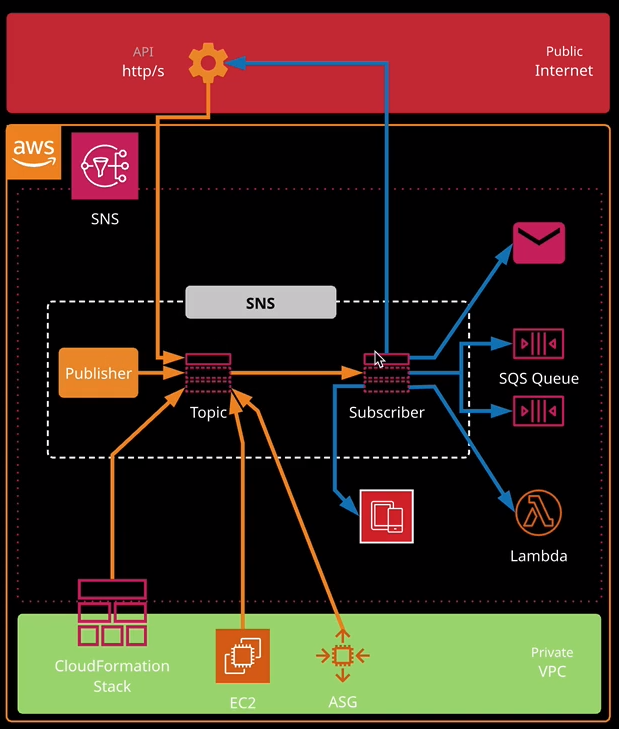
Now **SNS is a highly reliable product. It's designed to be a business critical notification service and so because of that, it's resilient across all AZs in a region. So it is a regional service.** When you create a topic inside SNS you're doing so in a particular region but **it is fully resilient to the failure of availability zones inside that region. It's a fully managed service and it can scale to whatever load levels are required. Now it is a public AWS service**. It's **got a public endpoint and so to access it from inside a VPC, you need either an internet gateway, a NAT gateway, or a VPC endpoint and you could also access it from any other location with a public internet connection.** So, any publicly connected internet host or any on-premises location that has got internet access. As long as you can reach the public endpoint, you can access SNS.

So if I move across to the SNS console, move to topics, create a topic, I'll need to name it. So I'll call it SNS lesson and I'll use the same for description. Now, **SNS does support both types of encryption. That's encryption at rest using KMS and also encryption in transit**. Now **with SNS, you could also define resource policies on the topic. So you can allow other AWS accounts to send messages to a topic and even allow everybody.** So the fact that you can do resource policies means you've got a lot more flexibility with the permissions on an SNS topic. So that's really important to understand for the exam, **you can use resource policies on the topic**. Now, in terms of the exam essentials messages that **are sent to an SNS topic could be up to 256 kilobytes in size**.

By default, any subscribers to that topic will receive all messages. So what do I mean by subscribers? Well, we know that a topic is something that you can send messages to and on a topic, you can create subscriptions. Now there are lots of things that can actually subscribe to a topic. You can subscribe HTTP or HTTPS endpoints. So API endpoints can receive a copy of any messages sent to the topic. You're able to subscribe email addresses, and they can either get raw emails or a JSON version of the data being sent to the topic. You can create an Amazon SQS subscription to that topic. So whenever a message is sent to the topic, a message is added to SQS queue and I'll talk about that more in the next lesson. A Lambda function can also subscribe to a topic. So when a message gets sent to that topic a Lambda function gets invoked and can perform some form of processing and then you're also able to subscribe SNS endpoint. So if you want to use this, maybe is part of a systems management or notification platform, then you can have engineers subscribed to a topic using SNS, and they'll get notifications when any messages are sent to the topic. So if I wanted to go ahead and subscribe my own email address to this topic, I could go ahead and answer my email address so I can scroll to IO and then create a subscription. I'd receive an email, direct my email account, and I need to confirm the subscription this is just to avoid spam. So confirmed subscription. That's that confirmed when we get that confirmed dialogue and at that point, any messages that I send to this topic, I'll get a copy of that in my email address so if I go to topics go to SNS lesson, publisher message. I'm going to keep the subject simple. This is a test, and then the message body I'm going to say hello, world. Now when I'm sending a message I can either define the an identical payload will be sent to all the protocols. So all of those subscribers, maybe some of them are email based. Maybe some of them are SNS based and by default, using this option, the same payload will be sent to all of the subscribers using those different protocols. I am able to pick this option, which allows me to define a different payload for different types of endpoints so I can define a default. I can define a message that any email endpoints get but I could also define different payloads, for example, for SNS, for HTTPD and HTTPS endpoints and a different one again for Lambda endpoints. So I've got a lot of flexibility. In this case, I want to keep it simple, so I'll create a simple hello world identical payload, and I'll publish that message. So when I publish a message, it gets sent to the topic, and then any subscribers to that topic will receive that message, and this is a simple example, but it should illustrate that this has been delivered as a service. I don't need to worry about the admin, the scalability, the security, the high availability. That's all handled by the product. Now, as well as the subscriber types that I have just talked about so HTTPD, HTTPS, email, Amazon, SQS, Lambda, and SNS.

SNS is also capable of operating as a mobile push platform so you can create a topic that's designed to send push notifications to all of the popular mobile platforms. Now another really powerful feature of **SNS is the ability to define filters for individual subscriptions.** So with a lot of notification platforms, whenever you send a message to a topic, every subscriber gets a copy of that message but imagine a scenario where you're running a notification system and you've got different tiers of engineers that have subscribed to this topic and you only want to notify certain groups of engineers when a message is sent to the topic that is relevant to that group where you can define filters for individual subscriptions and only have messages sent to those subscriptions when there are certain relevant fields in those messages. **So that means that the filter logic has been done on the topic rather than having to have every subscription receiving every message and then drop the ones which aren't relevant.** So that's a really powerful feature and it allows you to use one topic for lots of different functions.

Now, in terms of the Solutions Architect Associate exam, I want you to be comfortable with the concept of topics. **So a topic is an entity that you can send messages of up to 256 kilobytes to. I want you to be aware of what subscriber is. So subscribers to a topic of the things that going to receive those messages and these are the important endpoint type.** I've talked about those, but I would try to memorize those for the exams. So understand the type of subscribers that SNS is capable of working with. That's really important for the exam in terms of publishers, you'll see throughout this course and when using AWS in production, almost all AWS services are capable of sending SNS notifications to a topic. It is the de facto notification system inside AWS and because of the APIs and the SDKs, you can integrate SNS easily with any platforms that you architect and if you're developer you can easily code the functionality in direct to your application.



Now, architecture wise could be the foundation for lots of different types of architecture. So I've already mentioned how most AWS products use topics to send notifications but you can also use topics with other AWS services. Now I'm going to be talking about SQS queues in the next lesson but you can use a **topic for an architecture called SQS fan out** and it's a really important architecture to understand for the exam, I want you to imagine the situation where your architects in application, where people can upload videos to a video hosting website, something like YouTube. What you might do is when you upload a video, you might send a message to a topic. So straightaway this topic has a copy of the message, which indicates the raw media file that you've uploaded, and maybe that file's location on S3. Now what you might have a multiple queues subscribing to that topic. Maybe a queue for each independent bit rate that you want to convert that master or raw media file into. Well, let's say you have five queues that are subscribers to that topic. Well, every queue is going to receive an identical copy of the message that's sent to that topic. So each queue now has one message, and you can have those queues that are being polled by worker fleets of EC2 instances that are doing specific processing. So that's the way that **you can take one message that gets sent to an SNS topic and fan it out to multiple queues for further processing. So that's called SQS fan out** and you might well find that you get a question on that in the exam which talks about how you can have one upload or one event and have that fanned out to multiple destination systems and if you have any questions think about SQS fan out.