Dealing with Network Timeouts

In this lesson, you will learn how to deal with network timeouts.

WE'LL COVER THE FOLLOWING

- Check timeouts for all involved services

In the previous example, you were uploading relatively small files to S3, so that operation completes quickly. By default, Lambda allows a function just three seconds to finish. A process that is any longer will cause a timeout. If you plan to process larger blocks of data or perform many network requests within the same function, three seconds won't be enough.

You can increase the allowed time by setting the `Timeout` property in the function SAM template. This value is measured in seconds and the maximum you can set it to is 900 (15 minutes). To change the value of the `Timeout` property for all functions in a template file, modify the `Globals` section.

When a Lambda function is talking to AWS resources, you generally don't need to worry too much about availability and local access latency. When a function is communicating with third-party services, don't ignore potential network problems. Increasing the timeout is good but does not really solve the problem. If a Lambda function times out, API Gateway will just return a generic error to the clients, and they won't know what happened. It's much better to set up a circuit breaker and interrupt a stalled network process while you still have time to respond with a sensible error message to your clients.

The Lambda `context` object, which you used to retrieve a unique request ID, has a few additional interesting properties. One of them is the function `getRemainingTimeInMillis()`, which returns the number of milliseconds left for the execution of the current function. Use this information to schedule an event a few seconds before the end of the function when talking to potentially unreliable external services or performing long-running tasks in a Lambda function. You can then clean up and respond to the client before the external resource times out.

In fact, letting the Lambda run for longer might not make a lot of difference over half a minute for HTTP requests. API Gateway also has a timeout that will kill any requests lasting more than 29 seconds. There is no way to change this, but if you find yourself in a situation where that's a problem, you probably need to rethink the application architecture a bit. In the next chapter, you will look at some nice solutions for long-running processes.

> ## Check timeouts for all involved services
>
> When using multiple AWS services, remember that they might have different timeout settings, so you may need to configure all of them or work around their constraints. For example, API Gateway limits requests to 30 seconds, so setting a long timeout value for Lambda functions driven by API Gateway has no real effect.

You have reached the end of this chapter. Next, you have some interesting experiments waiting for you!