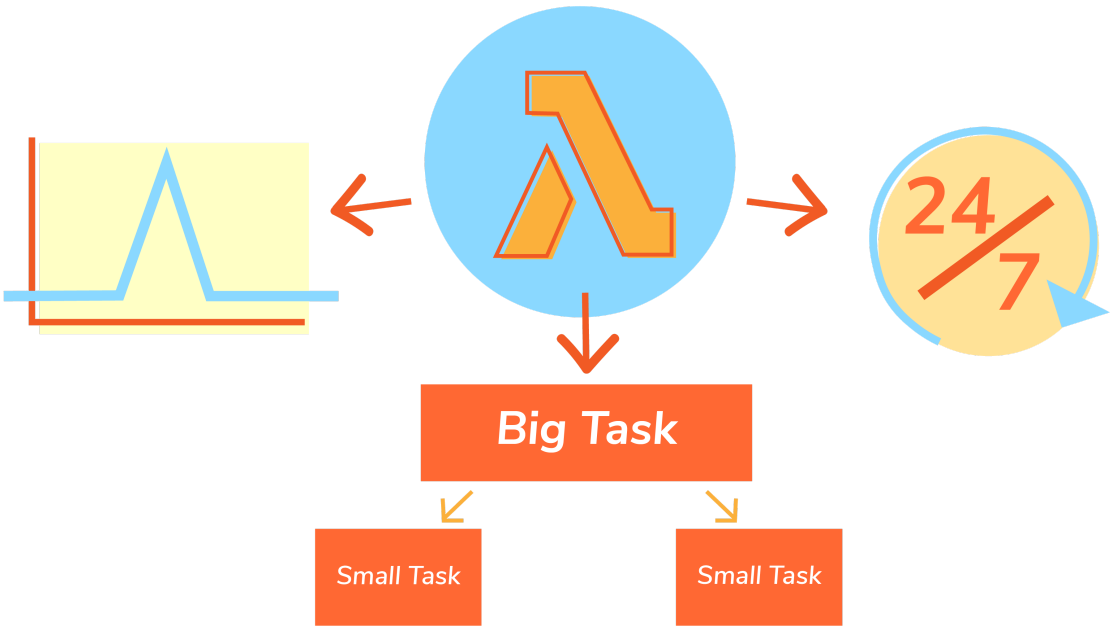
When to Use Lambda

In this lesson, you will learn when and when not to use AWS Lambda.

WE'LL COVER THE FOLLOWING

When to use Lambda



Maximising throughput

Lambda is great for use cases where throughput is critical and the tasks parallelise nicely. Typical web requests for dynamic content, involving access to a back-end database, or some user data manipulation usually fall into this category. Automatic email replies or chatbots are also a nice example. Any single request taking a few hundred milliseconds more than average won't be noticeable to typical users and Lambda will ensure that everyone gets served relatively quickly regardless of traffic spikes.

Splittable longer-running tasks

Longer on-demand computational tasks that can execute in less than 15 minutes, or could be split into independent segments that take less than 15 minutes, are also a good use case for Lambda. In these cases, the few hundred milliseconds required for Lambda functions to start won't make an important difference to processing time. Some nice examples of tasks that fall into this category are file format conversions, generating previews or thumbnails, and running periodic reports.

High availability tasks

Tasks that need high availability and good operational infrastructure, but do not need to guarantee latency, are also a great use case for AWS Lambda. This includes payment notifications from external systems, for example, PayPal or Stripe. These notifications must be handled reliably and relatively quickly, they might have unpredictable traffic patterns, and yet it's not critically important if they are finished in one second or two seconds.

When not to use Lambda

Latency guaranteed tasks

Lambda is currently not suitable for tasks that require guaranteed latency, such as in high-frequency trading systems or near-real-time control systems. If a task must be handled in under 10 or 20 ms, it's much better to create a reserved cluster and have services directly connected to a message broker.

Longer-running tasks

Another category where Lambda isn't suitable right now is tasks that could

potentially run for longer than 15 minutes. One notable example is video transcoding for large files. Connecting to a socket and consuming a continuous data feed is also not a good use case for Lambda, due to the time limit.

Tasks demanding high processing power

The third category where you should not use Lambda right now is tasks that require a huge amount of processing power and coordination. For example, video rendering. Tasks like that are better suited to a reserved infrastructure with a lot of CPUs (or even GPUs).

Tasks requiring no on-demand computation

Lastly, tasks that require no on-demand computation, such as serving static web files, are a poor use case for Lambda. In theory, it's possible to use Lambda as a web server and send images and CSS files to clients, but this is a waste of money. It is much cheaper and faster to use a specialized product for that, for example, a content delivery network.

Conclusion

A nice aspect of Lambda is that it works well with all the other services in AWS, so it's possible to incrementally adopt a serverless approach for certain tasks and use more specialized AWS products for tasks where Lambda does not fit nicely. You can read about some typical ways to combine Lambda with other services in the next chapter.

Starting in the next chapter, you'll build a service that combines the first two categories of tasks from this section: web processing with background file conversions.

>  If you want to develop and test serverless applications locally (on your machines), you'll need to set up some tools for this. You can follow the setup instructions mentioned in the next chapter. However, in this course, you already have the configured environment for you to run the code and the command lines in your widget. Therefore, you don't need to worry about the setup.

The next lesson lists some interesting experiments. You can perform these to get a better understanding of what you have seen so far.