How Request Pricing Affects Product Decisions

Let's see how request pricing (i.e, charging customers based on the number of requests) affects product decisions.

WE'LL COVER THE FOLLOWING

- Testing with AWS Lambda
- Understanding the flow of capital on a granular level

Lambda pricing can also change how organisations release features. Say you have a customer with some very specific needs. You can implement a new feature for them in a few days, but it would take weeks more to build it up to work for all your other customers. With Lambda, you can launch a new version for the specific customer as soon as the feature is ready, and keep everyone else on the old version until the feature is fully developed. Running two environments doesn't cost any more than a single one.

Testing with AWS Lambda

Similarly, request-based pricing opens up new possibilities for testing features. Do you want to test how a change in a back-end process affects user conversion? Just create an experimental version of the application and send 10% of the traffic there. The number of users and requests is still the same, so running two versions costs the same as working with a single version. Running A/B split tests on the front end is easy, but organising that kind of work on back-end features was traditionally very expensive. Five years ago, only companies with massive computing resources, such as Google or Facebook, could run experiments on their back-end code continuously. With Lambda, that's now available to everyone, even a single-person team. It doesn't cost anything more than running a single version.

Understanding the flow of capital on a granular level

Another interesting aspect of the new pricing model is granular cost breakdown, down to every single request. For AWS to charge you correctly, it accounts for each request, and this information is also available to you as a customer. So it's now possible to understand the flow of capital through the application at a level of individual operations. You can calculate exactly how much money a single customer is costing you to serve, and decide whether it's worth keeping them around or not. You can calculate exactly how much money certain features cost to operate, and then decide whether they are worth enhancing or perhaps removing.

I hope, by now, you have a good idea about the pricing model of AWS Lambda. Let's discuss the important technical constraints of AWS Lambda in the next lesson. See you there!